PC-8801mkII グラフィック・ワークブック

桜田幸嗣・蓑島 聡 共著

ASCII
アスキーブックス

PC-8801mkII

Graphic Workbook

グラフィック・ワークブック

ビジュアル体験で学ぶBASICプログラミング

桜田幸嗣・蓑島 聡 共著



PC-8801mkII
PC-8801
対応

アスキー出版局



Designed by Kenichi Samura／Number One Design Office.
Photographed by Taishi Hirokawa

PC-8801mkⅡ

Graphic Workbook

グラフィック・ワークブック

ビジュアル体験で学ぶBASICプログラミング

桜田幸嗣・蓑島 聡 共著

アスキー出版局

本文デザイン……宇治　晶
本文イラスト……東恩納　裕一

イントロダクション

PC-8801mkIIグラフィック・ワークブックをお届けします.

最近のパソコンは非常に豊富な機能を備えるようになってまいりました. 特にグラフィックの高解像度化, 表示カラーの多様化には, 目を見張るものがあります.

そのような高性能マシンが数あるなかでも,PC-8801mkIIはひときわ優れたグラフィック機能を誇っています. これを存分に使いこなせば実に多彩な表現が可能です.

けれども, あなたはそれを単なる可能性のままで終わらせてはいませんか. 次のような悩みをかかえてはいませんか.

1. グラフィック機能を使って, いったい何をすればいいのかわからない(目的不明)
2. やりたいことはたくさんあるけれど, プログラムが作れない (力不足)
3. グラフィックスはどうもめんどうで, 手が出せない (気力減退)

本書を読んでいただきたいのは, まさにそのような「あなた」です.

目的が見つからない方には数々の面白い題材を, プログラムがわからない方にはその手段と方法を, そして今までグラフィックスに関心がなかったという方には新鮮な興味を, 我々は提供したいと思います.

そうです. 本書はコンピュータ・グラフィックス入門書としての体裁を整えてはおりますが, 決してそれを唯一の目的としているわけではありません.

BASICの基礎知識を身につけ, 次なるステップへいざ踏み出そうという方にとっては, 本書のプログラムを読み, かつ理解することがプログラミングのよき練習となるでしょう. そのような用途にも役立ててもらえるよう, 本書のプログラム解説は紙面の許すかぎり詳細に, 初心者の方にもわかりやすく行っております.

また適当にページを開き, そこに掲載されたプログラムを実行してみる, というのも本書の楽しみ方のひとつです. 順を追って読んでいき, グラフィックスの知識を積み重ねるのも結構ですが, 眠れぬ夜の退屈しのぎにグラフィックスでも……というのも,なかなかよろしいではありませんか.

コンピュータ・グラフィックスはたいへん楽しいものです. そしてそれを自らのプログラムで描き出す喜びはまた格別です. 本書を通して, この楽しみをより多くの方に味わっていただければ幸いです.

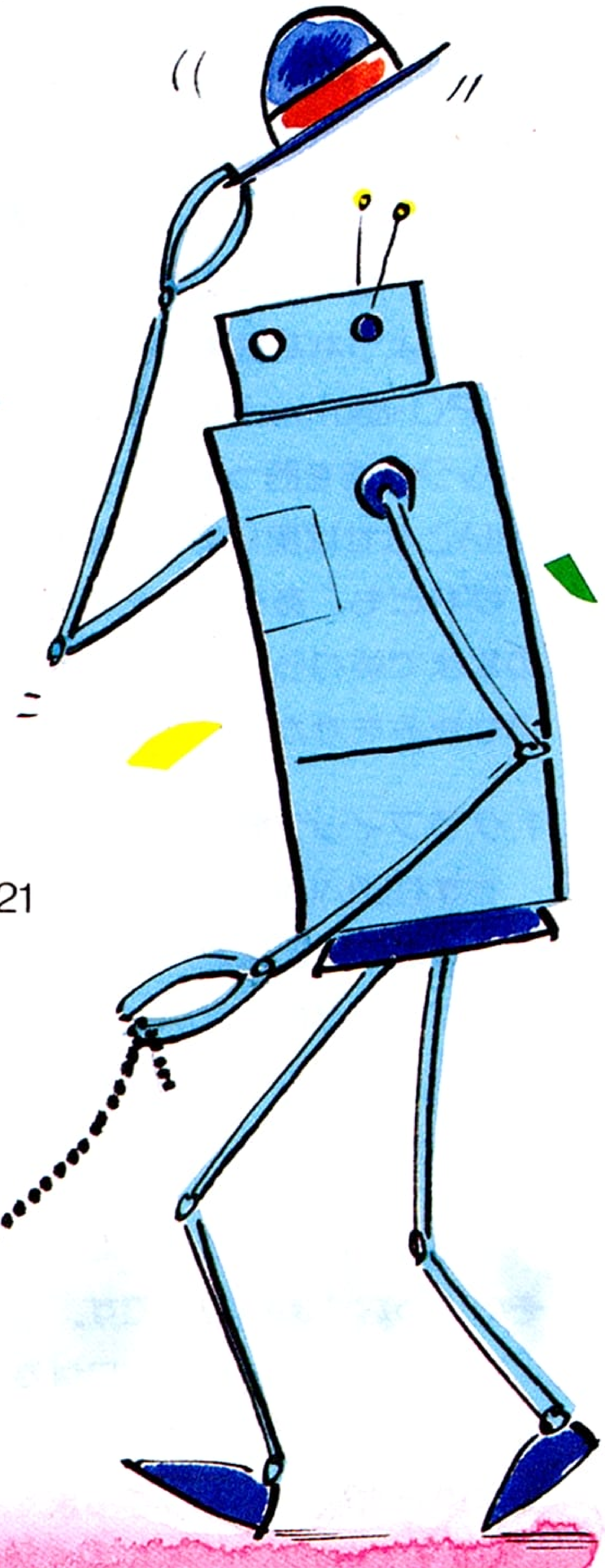
桜田 幸嗣
蓑島 聡

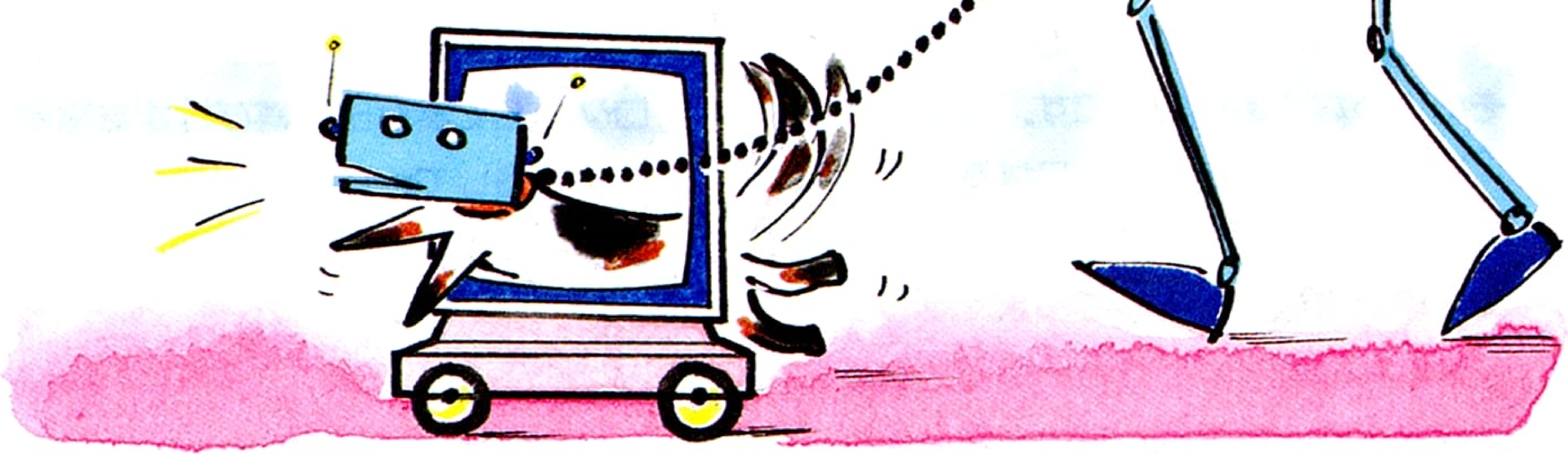
PC-8801 mk II
グラフィック・ワークブック



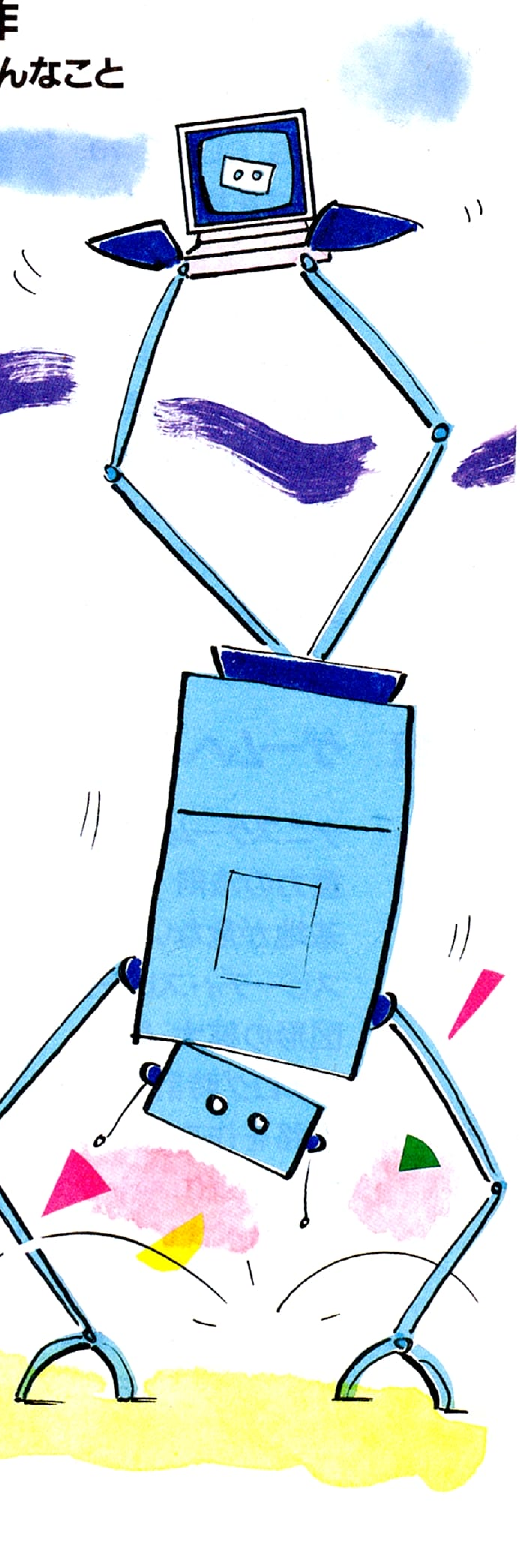
【目次】

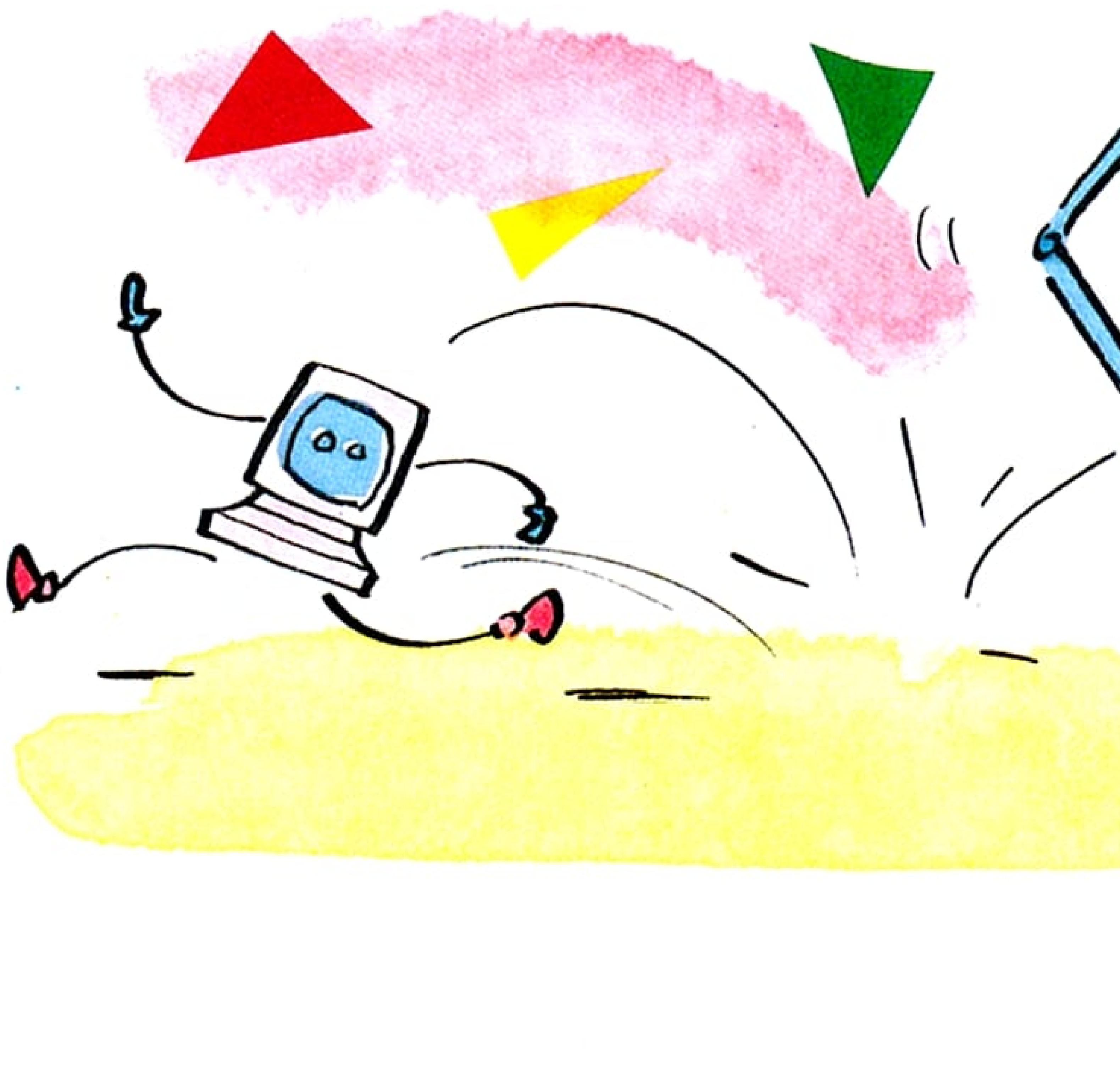
1章　忘れていませんかグラフィックの約束ごと PC-8801 mk IIの使い方





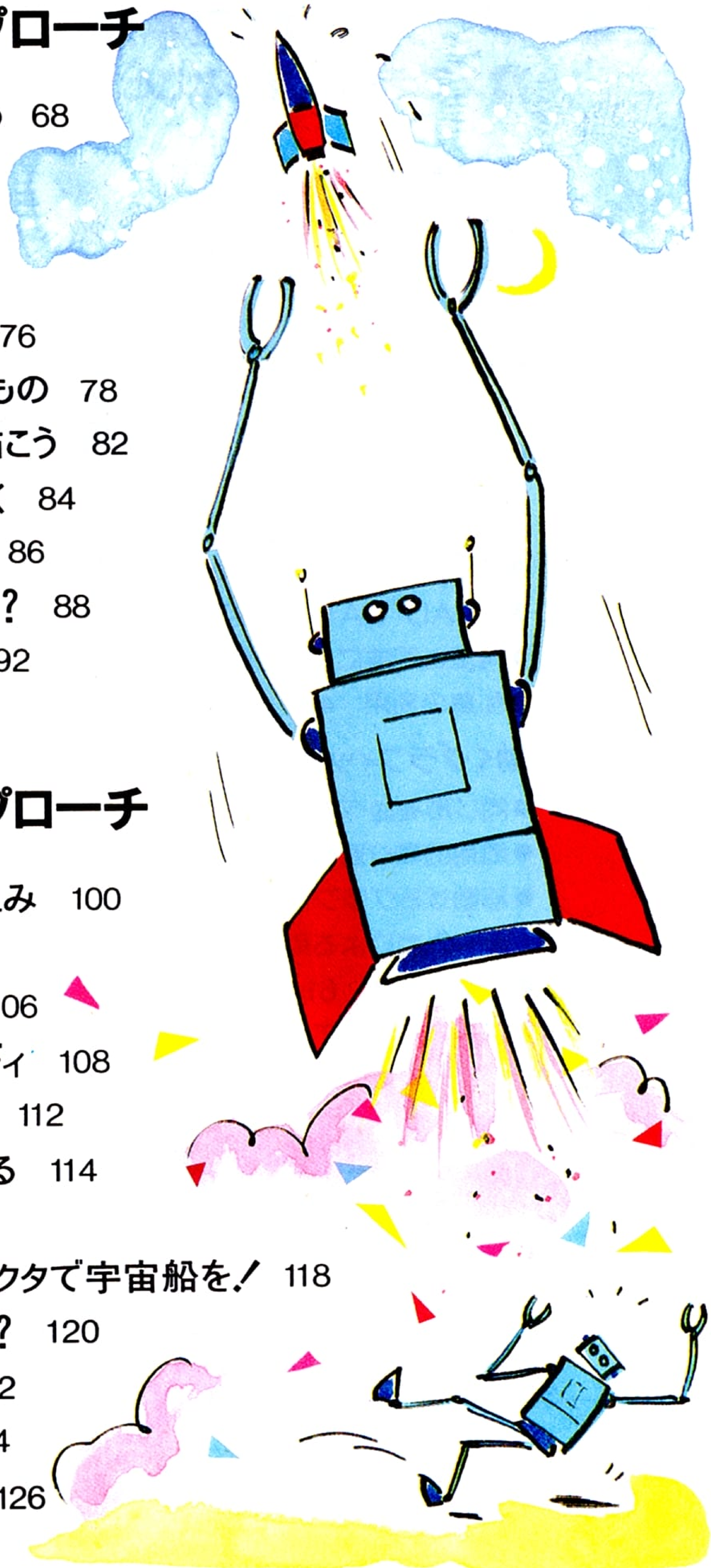
2章　グラフィックの基本操作
覚えてほしい，あんなこと，こんなこと



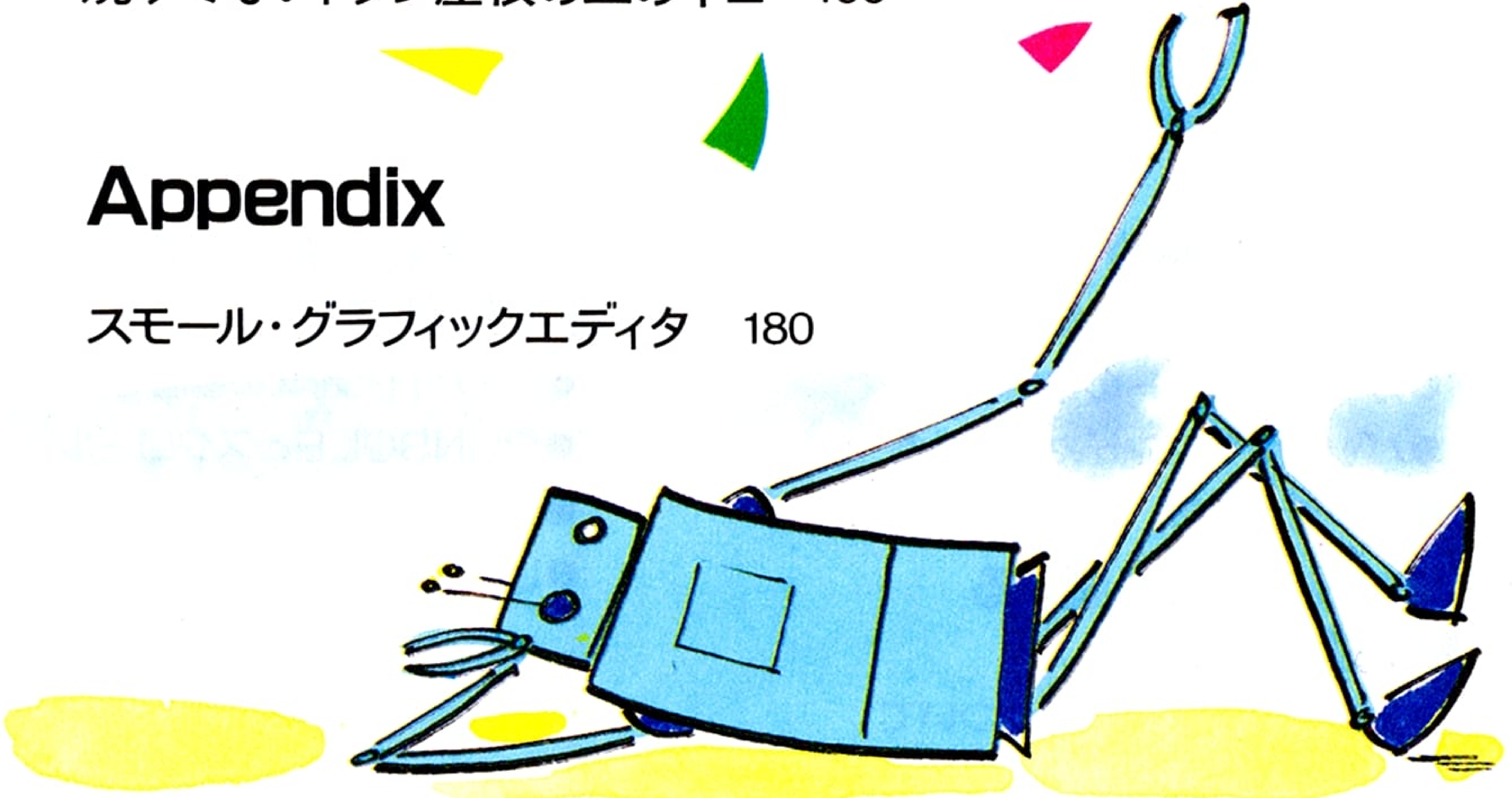


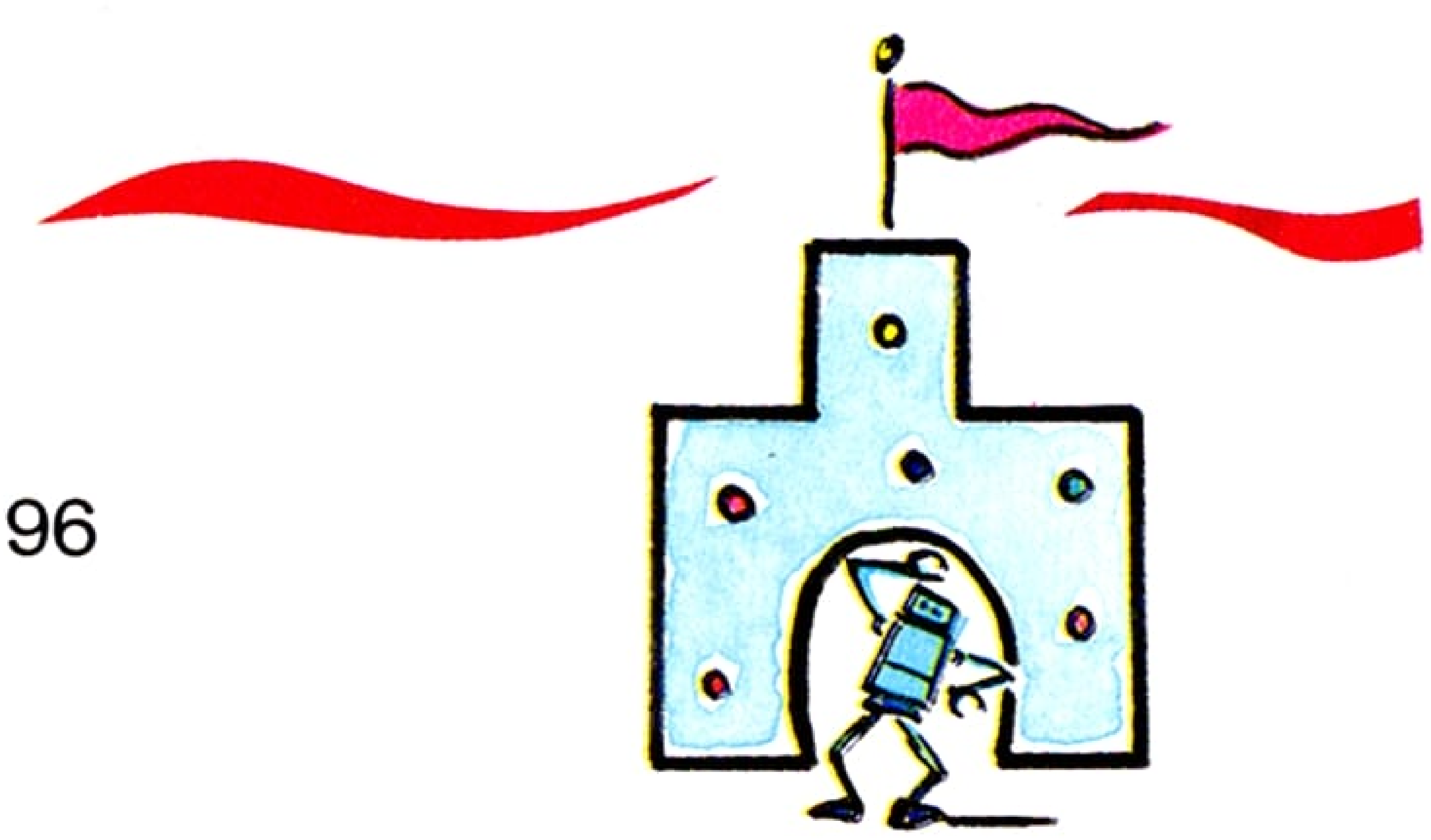
3章　アートへのアプローチ

4章　ゲームへのアプローチ









【1章】

忘れていませんかグラフィックの約束ごと
PC-8801mkIIの使い方

グラフィックを始める前に

- ●画面の種類とその設定
- ●カラーパレットの初期設定
- ●バックグラウンドカラー推参
- ●その他の設定

グラフィック命令に関する基本的御注意

- ●PSET，PRESETについて
- ●LINEについて
- ●CIRCLEについて
- ●GET，PUT
- ●PAINTについて

キャラクタ表示命令に関する基本的御注意

- ●WIDTHと趣味の画面モード
- ●CONSOLEとスクロール
- ●COLORについて
- ●LOCATEによる位置の指定



コンピュータ・グラフィックスの世界へようこそ．まずは手慣らし足慣らし，PC-8801 mk IIのグラフィック機能について，いろいろな話題を取り上げてお話しいたします．

実は企画段階では，本章はグラフィック手帳といいますか，個々のグラフィック命令の使い方を解説したマニュアル的雰囲気のページにしようと考えていたのです．けれども，原稿を書き進めていくうちに，筆者の心はむなしさで一杯になってきました．

こんなこと書いても，皆 PC-8801 mk IIのマニュアルに載っていることばかりではないか．同じことをここで書いたからって，読者の方には何も目新しいことはないんじゃないか．

そうなんです．だいたい，1冊の本の中でありとあらゆる説明を行おうとすることに，ムリ，ムダ，ムラがあるのです．ただでさえコンピュータ・グラフィックスには話のタネが多いもの．あれもこれもと欲張っていては、いつまでたっても本が終わりになりません．各命令の使用法に関しては，既に立派な PC-8801 mk II BASIC マニュアルというものが存在しています．これを見ていただけば十分ではありませんか．

というわけで，本章ではマニュアルをそのまま書き写したような説明は極力避け，それ以外の有益な小知識，あるいはコンピュータ・グラフィックスに関連したヨタ話などを，いくつか紹介いたします．どうぞ気軽に読み流してください．ご参考になる部分が1か所でもあれば，満足であります．

グラフィックを始める前に

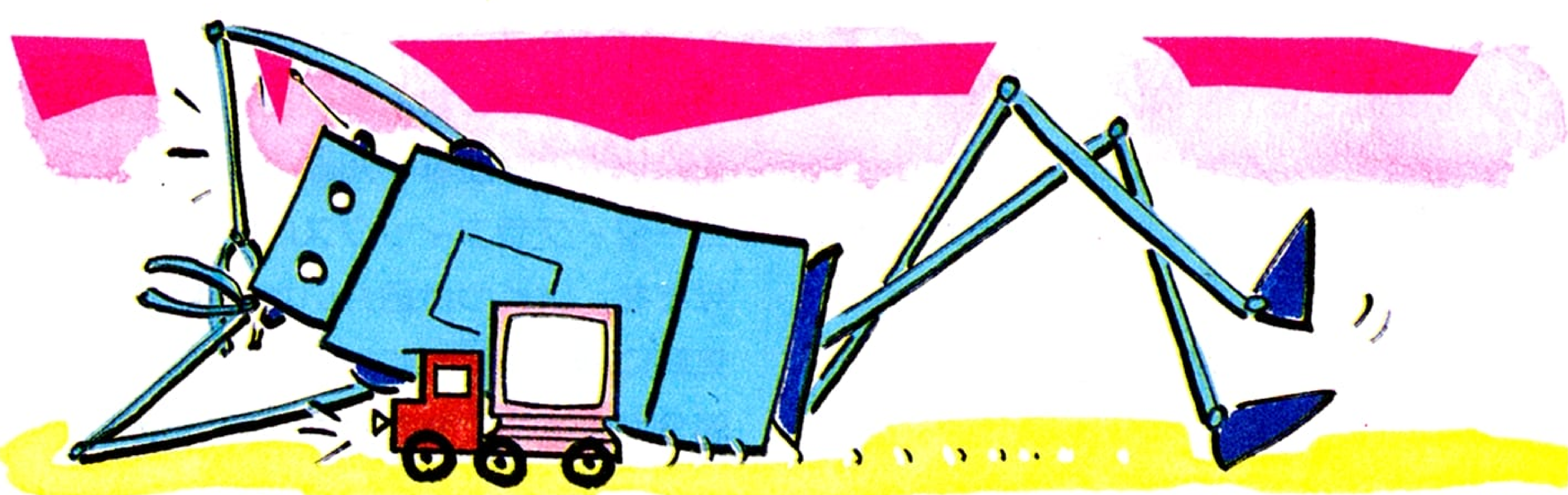
おや，おかしい．プログラムは絶対に正しく打ちこんだよ，ねー．何度見たってマチガイないでしょ．なのになぜ写真と同じ絵が出てこないんだろう．うわぁーん．ボクのPC-8801 mk IIが壊れちゃったよぉ．——などとお嘆きの貴兄に．あれ，そこのあなた，ヒトゴトのような顔しないでください．誰でもこういう事態におちいるかもしれないんですよっ．

というわけで，自分のPC-8801 mk IIがどこかおかしいのでは，と心配しているあなた．いやいやパソコンなんてものは案外と丈夫にできているものです．めったに壊れることなどありませんからご安心を．

正しいはずのプログラムが期待したように動かないときは，どこか画面表示に関する設定が変わっていると思って間違いありません．机に向かってPC-8801 mk IIのスイッチを入れて，さぁこれからプログラムを作るぞぉー，という場合はよいのです．そんなときは本に載っているリストどおりに打ち込めば，まずトラブルは起こりません（もちろん正しく入力できていれば，の話ですが）．けれども，中には興じていたゲームが一段落して，いざ本でも読もうか，などと本書を手にする方もいるでしょう．予期せぬできごとが発生するのはそんなときです．

いわく，色がおかしい，文字が表示できない，などなど．それらは前に実行したプログラムによって，画面表示に関する設定が，いつもとは異なってしまったものがほとんどです．

このようなグラフィックの表示に関連したトラブルの原因とその解決法を，まとめてめんどうみようじゃないか，という次第で本節が設けられました．



画面の種類とその設定

PC-8801 mk IIには，テキスト画面とグラフィック画面の2つのモードがあります．別に，ディスプレイが2つ心要なわけではありませんから，オマチガエのないように．テキスト画面はリストの表示やプログラムの打ち込みの際に利用している画面です．グラフィック画面は，LINE命令やCIRCLE命令を使って，いわゆるグラフィックを表示する画面です．PC-8801 mk IIでは，テキスト画面がグラフィック画面の上に重なって，同時に表示されます．つまり既に表示されているグラフィックを消してしまわなくてもリストを重ねて表示することができるので，グラフを描いたり図形を出力させておいてリストを見るときなどは，大変便利であるといえます（他機種のマシンの中にはテキスト画面がなく，文字やグラフィックシンボルもグラフィック画面に表示するものがあります）．

テキスト画面については後で述べることにして，ここでは，これから本書で利用していくグラフィック画面について考えてみることにしましょう．

PC-8801 mk IIのグラフィック画面は，3種類の画面モードを選ぶことができます．これはSCREEN命令で指定します．書式は以下のとおり．

```
SCREEN 〈画面モード〉,〈画面スイッチ〉
        └第1パラメータ  └第2パラメータ
```

第1パラメータは，0～2の3種類の画面モードに対応しています．それぞれのモードには，*Figure 1* に示すような性質があります．

SCREEN 0を指定してカラーモードを選択すると，美しいカラーグラフィックが可能になります．本書では主にこのモードを使ってグラフィックの基本テクニックを説明していきます．他のモードは本書では利用しませんが，簡単に説明しておきます．SCREEN命令には，他に第3，第4パラメータによる指定があります．SCREEN 1で白黒モードにすると，この第3，第4のパラメータを指定して，異なった白黒画面を最高3枚まで扱うことが可能になります．異なる3種類のグラフを比較するようなプログラム，あるいはコンピュータアニメなどでこのモードを利用すると便利です．次に，SCREEN 2で高分解能白黒モードにすると，カラーモードに比べてよりきめの細かい図形を描くことが可能です．また，漢字ROMを使って，漢字を出力させるときは，細かな線まで表示可能なので，たい

	モードの名称	ドットの構成	モードの性質
SCREEN 0	カラーモード	640×200ドット	1ドットごとに8色，色指定が可能
SCREEN 1	白黒モード	640×200ドット	モノクロ指定のみ(3枚)
SCREEN 2	高分解能白黒モード	640×400ドット	モノクロ指定のみ

Figure 1

ていこのモードを使います．ただし，高分解能白黒モードは，専用の高解像度ディスプレイを装備しないと利用できません．

PC-8801 mk IIのグラフィック画面の3つのモードについて説明してきましたが，グラフィックを行う前には，SCREEN命令で必ずこのモード設定を行うようにしてください．

SCREEN命令の第2パラメータについて，少し話しておきましょう．このパラメータでは，画面スイッチの設定をします．画面スイッチには，グラフィック画面を表示するかどうかを指定するスイッチと，グラフィックを高速に出力するかどうかを指定するスイッチの2つがあります．詳しくはマニュアルを見ていただくとして，本書では，通常は0に設定しています．1を指定すると高速書き込みスイッチがONになりますので，通常よりも画面への書き込みが高速になります．ただし，画面がちらつきます．

本書では，より美しいグラフィックを描くことを主たる目的としているため，以後はSCREEN 0，0で設定したカラーモードのグラフィック画面を利用していくことにします．

カラーパレットの初期設定

PC-8801 mk IIのスイッチを入れた時点では，0から7までの番号が，順に，黒青赤紫緑水黄白の各色を表示するように設定されています．この番号と色の対応をプログラム中で変更できるというのが，カラーパレットの機能です．

この機能は，特にグラフィックを高速に移動させる場合に，たいへん有効なものです（本書の4章でもこのテクニックを紹介していますし，市販のプログラムにも，カラーパレット切換えの機能をうまく利用したゲームがあります）．

しかし，パレット機能の使用に際しては，後始末が大切です．変更したパレットをそのままにして別のプログラムを実行しようとすると，思いがけない結果が生ずることもあるからです．

例えば，赤い線を引こうとして2のパレットコードを指定したら，たまたまパレットコード2には前のプログラムのなごりとしてカラーコード0の黒が割り当てられていたとします．この場合，画面には何も現れません．驚いてプログラムを見直しても間違いはない．うーん，と頭を悩ますわけです．

特に本書の中では，一部を除いて，カラーパレットの設定は電源スイッチを入れた時点から，何の変更もないことを前提にプログラムが作られています．ですから，もしカラーパレットの設定を変更させてしまった，あるいはそのような恐れがある場合には，下のプログラムを実行して，パレットを初期設定し直してください．

カラーパレットを初期状態に戻すプログラム

```
                                 パレットコード
FOR  I=0  TO  7:COLOR=(I,I)
:NEXT I
                                 カラーコード
```

バックグラウンドカラー推参

PC-8801 mk IIではグラフィック画面においてバックグラウンドカラーの設定ができます．マシンのスイッチを入れたときの，バックグラウンドカラーは黒に設定されていますが，

```
COLOR,1
```

を実行すれば，これ以後グラフィック画面のバックグラウンドカラーは1，すなわち青に設定されます．

ここで注意しなければならないのは，このCOLOR命令では単に色の指定を行うだけで，実際に画面の背景が青に変わるのはCLS 2もしくはCLS 3を実行したときである，ということです．

バックグラウンドカラーの指定によって背景に色をつけることと，PAINT命令で画面を青く塗りつぶすことは，まったく意味が違います．

PAINT命令やその他のグラフィック命令は，画面に色を塗っていく，というものであるのに対して，CLS 2，CLS 3やPRESET命令では，画面上に塗ってある色を取り除き，そこに開いた穴からバックグラウンドカラーが現れるということなのです．

バックグラウンドカラーを指定するということは，この開いた穴から見える下地の色を決めるということです．そこで，バックグラウンドカラーは，見苦しくないものを設定しておくことをお勧めいたします．

その他の設定

読者であるあなたが，自分でプログラムを組むとき，どうも思いどおりにならないという事態がしばしば起こることはありませんか？ 特にグラフィックを行う際には思った色が出なかったとか，図形がまったく表示されなかったとかいろいろなことがあるものです．そんなときは，今まで述べてきた種々の設定をもう1度確かめてみてください．それでもうまくいかなかったときは，画面の症状に従って，以下の治療を施します（編集者注：筆者は医者のタマゴなんで，あしからず）．

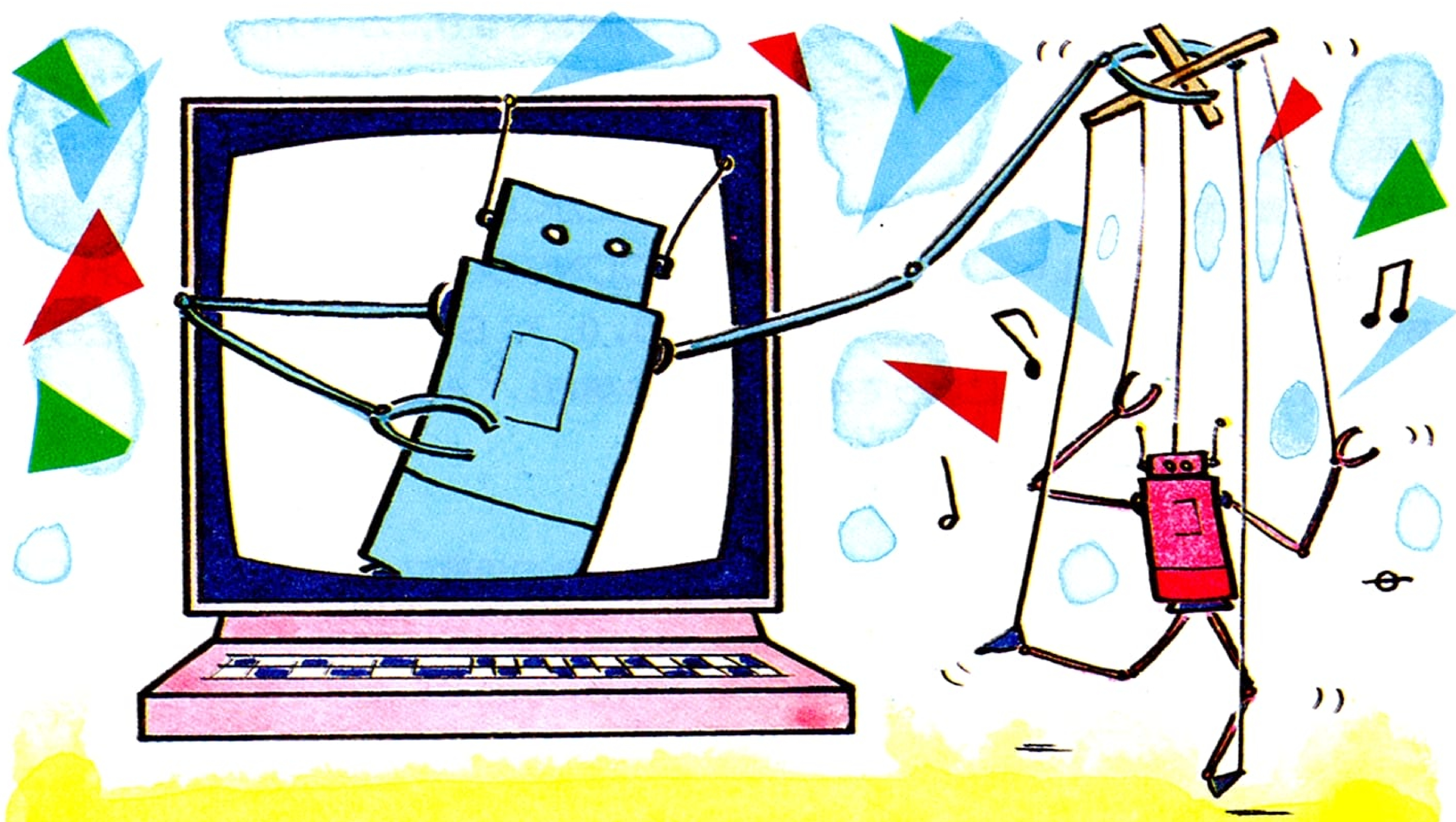
- 座標がおかしい： SCREEN命令で画面を再設定してみる
- カーソルがない： これはテキスト画面の問題ですが，カーソルがなければプログラムも打ち込みにくいはず．こんなときはLOCATE ,,1を実行してみる
- とにかくおかしい： WIDTH 80,25:CONSOLE ,,1,1:COLOR 7,0,0,7:SCREEN 0,0:CLS 3を実行してみる

これらの処置によっても，いっこうに事態が改善されない場合はきっとプログラムがおかしいのでしょう．もう1度リストを眺めてみてください．それでもだめなときは，マシンに向かって“ばかばかばか！”と叫んでリセットボタンを押しましょう．

グラフィック命令に関する基本的御注意

グラフィックの基本となる命令は，0次元の図形（"点" のことをもっともらしくいってるだけ）を描くPSETから1次元の図形を描くLINE，そして2次元の広がりを持つ図形を画面に一気に表示してくれるというPUT命令まで，各種とりそろえてあります．なんだ？ 3次元図形は描けないのか？――まぁ今に立体テレビでも普及したら考えてみましょう．じゃ4次元図形は，ですって？ あなた，この本はSF小説ではなくて，真面目なグラフィックの教科書なんですよ（いいすぎかな？）．

――それはさておき，これらのグラフィック命令の使い方は，皆マニュアルに詳しく載っていますし，それをくどくどとここでむし返すのはやめておきましょう．ただ，これらの命令によって何ができるのかということ，また各命令を使う際に少しばかり注意しておいてもらいたいことなどを，いくつか取り上げて解説してみようというのが本節の目的です．こんなことわかってる，という方は読みとばしてください．



PSET, PRESETについて

PSET命令は, 指定した位置に指定した色で, 点を1つだけ打つ命令です. PC-8801 mk IIに限らず, パソコンのグラフィックはすべて点（ドットともいいます）を最小の単位として描かれます. ですからグラフィックを描くには, 線を引くのも絵に色を塗るのも, 原理的にいえばPSET命令さえあれば何もいらないわけです.

しかし, これでは大きなグラフィック作品を描くことはまず不可能. そのため, 線を引くにはLINE, 色を塗るにはPAINTなど, それぞれ便利な命令が用意されているのです. PSET命令は, 部分的にちょっと点でも打ってみようかという場合にのみ使いましょう（などと書いてしまうと, わたしは点描派グラフィック画家だとかいう人が現れて, PSET命令だけで描いた「PC島の午後」なんていう大作を発表したりしてね. いえ, それはそれで賞賛いたしますよ).

本書の中で使っているPSET命令は,

```
PSET (x,y), パレットコード
```

というように命令中に必ず色（パレットコード）の指定を含めています. もちろん色は, COLOR命令でフォアグラウンドカラーを決めておき, PSETなどの命令の中では色指定を省略してしまう, という方法も可能なのですが, 本書では個々のグラフィック命令の中で, 色指定を行うという方針をとります. リストの打ち込みは多少めんどうですが, この方が, 確実にプログラムを理解していただけると思ったのです.

PSETが点を打つ命令であったのに対して, PRESET命令は指定位置から点を取り除くという働きをします. その結果下地であるバックグラウンドカラーがその位置に現れることになります.

なお, PC-8801 mk IIのグラフィック命令は, 座標を（x, y）といったように指定する代わりに, STEPによって相対的に座標を指定することが可能です. これは, 現在基準になっている点から横へいくつ, 縦へいくつ移動した所までという具合に位置を指定するものです. PSET命令でこの相対座標を使うときの書式は, 次のとおりです.

```
PSET STEP (x,y), パレットコード
```

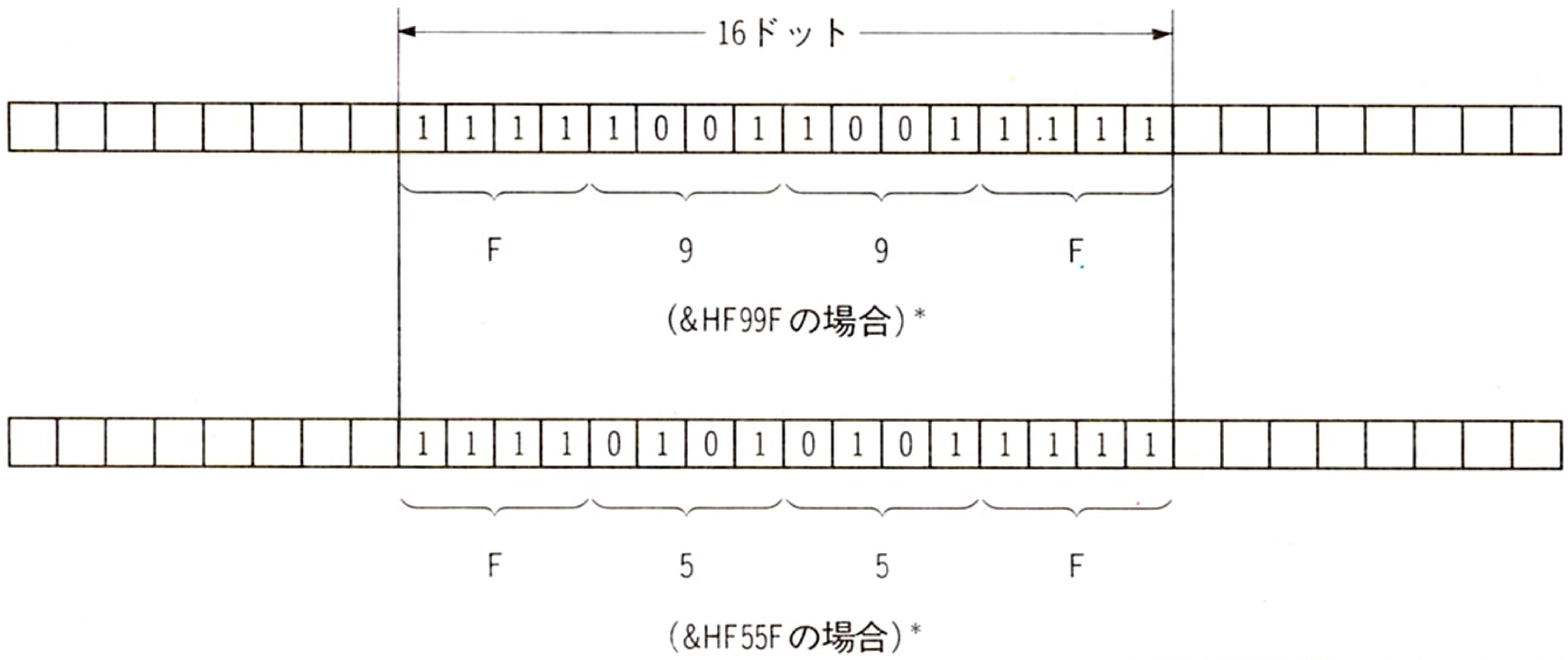
他のグラフィック命令でも同様に相対座標の指定ができます.

LINEについて

2点を結ぶ直線あるいはその直線を対角線とする四角形を描くLINE命令は, 次のような書式で表します.

$$\text{LINE } (x_1, y_1) - (x_2, y_2), \text{パレットコード}, \left|\begin{matrix}\text{B}\\ \text{BF}\end{matrix}\right|, \text{ラインスタイル}$$

この中で,「ラインスタイル」は, LINE命令で描く直線の型を設定するパラメータで, 通常は16進数によって表します. その設定は, まず16ドットを1組としてパター



*&Hは16進で表記していることを示す

ラインスタイル

Figure 2

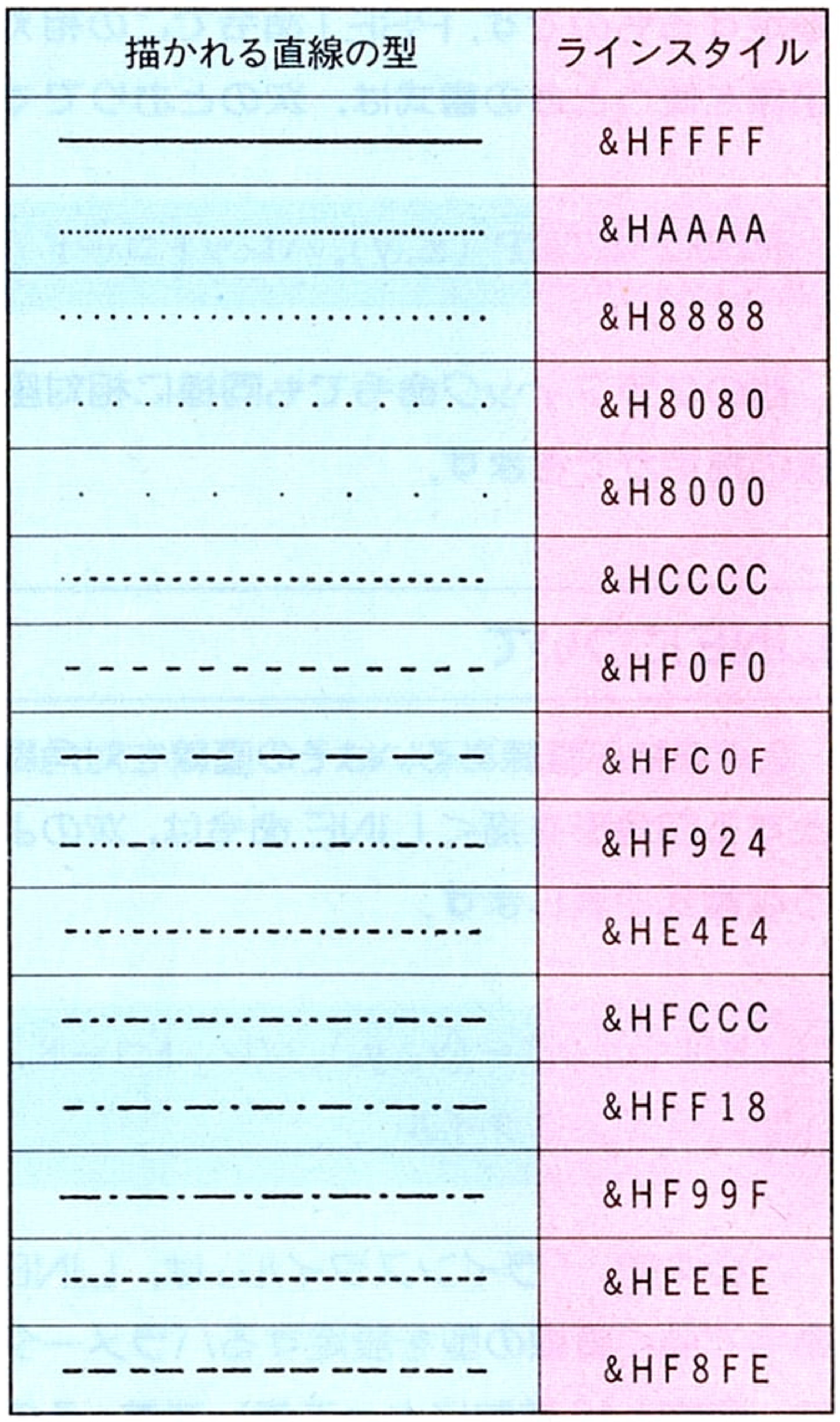


Figure 3

ンを作ります．その1組の16ドットを左から4ドットごとに4つに分け，それぞれの4ドットを2進数の0または1とみなして16進数に変換します．最終的には4桁の16進数になり，ラインスタイルが決まります（*Figure 2* 参照）．*Figure 3* にいろいろなラインスタイルの例を示しましたので参考にしてください．

CIRCLEについて

円または楕円あるいはその一部分を描くCIRCLE命令は，グラフィックには欠かせないたいへんありがたい命令です．円を描くのは簡単なようにも思えますが，これが案外難しい作業でして，もしCIRCLE命令がなかったら，プログラムに要する労力は何倍にも増えてしまうでしょう．

CIRCLE命令の書式は，次のように，指定できるパラメータがとても多く，これら

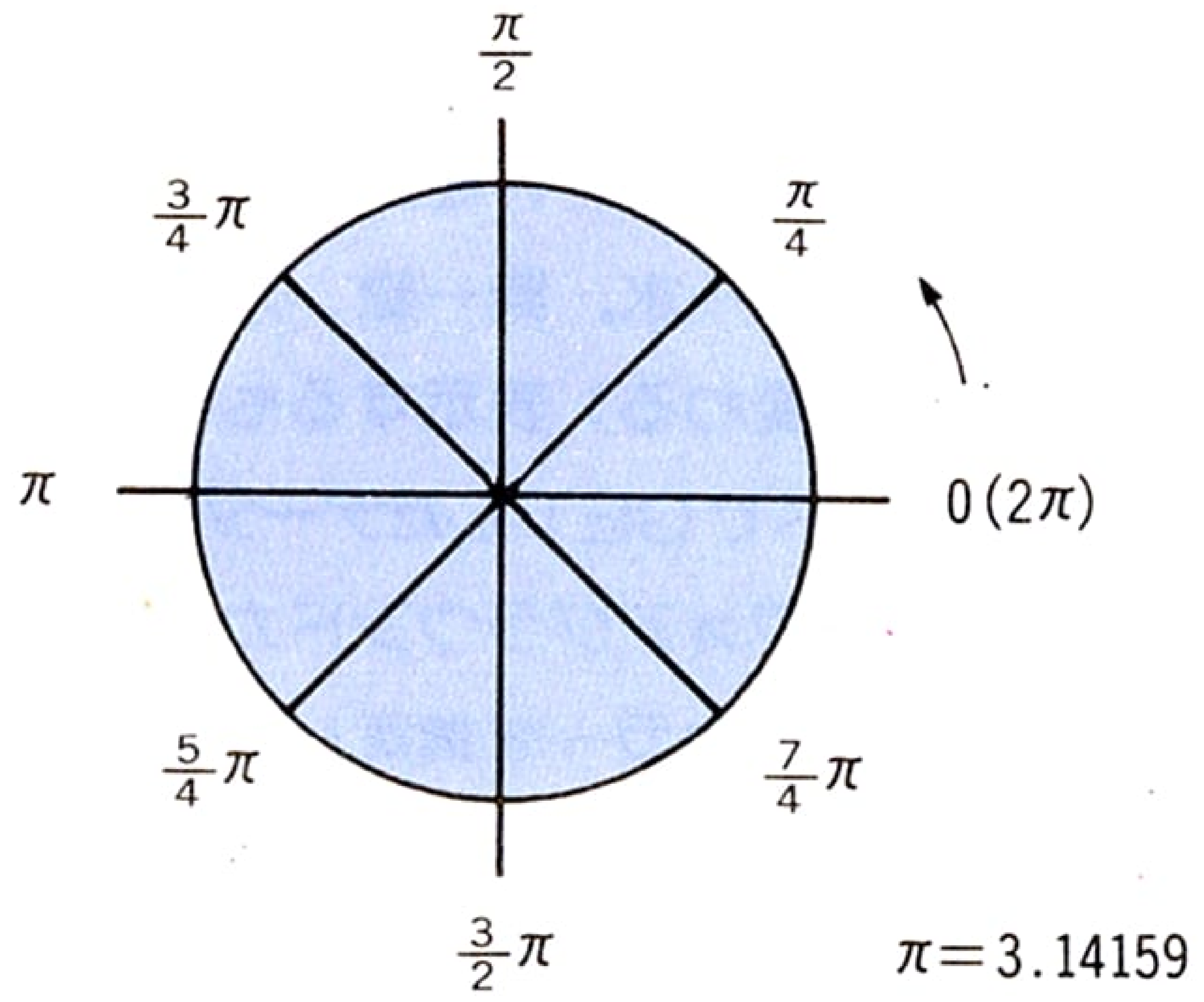


Figure 4

の使い方はなかなか覚えきれないかもしれません．そこで簡単に説明します．

CIRCLE (x,y),半径,パレットコード,
開始角度,終了角度,比率

(x，y)で中心の座標を指定し，半径は，x 軸方向のドット数で指定します．パレットコードは，円の色を示します．

比率は，円の x 軸方向の半径に対する y 軸方向の半径の比で，大きくなるほど縦長の楕円が描けます．

開始角度と終了角度は，円の１周を 0〜2π の範囲で，*Figure 4* のように与えた数値で，開始角度と終了角度の間の円弧が描かれます．このとき開始位置または終了位置を負の値で指定すると，その絶対値をとって円弧を描いたうえで中心とその位置を結びます．つまり開始位置と終了位置の両方を負の値にすれば，扇形を描くことも簡単というわけです．

GET, PUT

GET 命令および PUT 命令は，画面のデータを配列に取り込み，またそれを異なる位置に出力するために使います．GET 命令の書式は次のとおりです．

GET $(x_1,y_1)-(x_2,y_2)$,配列名
(要素ナンバー)

GET 命令の始点の座標には相対座標は使えません．

この GET 命令を用いる場合はグラフィックデータをすべて取り込める大きさの配列を，あらかじめ用意しなければなりません．必要な配列の大きさは，次のように計算します．

(((X+7)¥8)*Y*M+4)¥N+1

X：横ドット数　Y：縦ドット数
M：白黒では 1，カラーでは 3
N：整数型配列　2
　単精度型配列 4
　倍精度型配列 8

この GET 命令で読み込んだグラフィックを画面に表示するのは，PUT 命令です．その書式は次のとおりです．

PUT (x,y),配列名(要素ナンバー),条件,
フォアグラウンドカラー,バックグラウンド
カラー

ここで指定できる条件は PSET，AND，

		ビットパターン		
		緑	赤	青
黒	0	0	0	0
青	1	0	0	1
赤	2	0	1	0
紫	3	0	1	1
緑	4	1	0	0
水	5	1	0	1
黄	6	1	1	0
白	7	1	1	1

各カラーに含まれるもの ……1

各カラーに含まれないもの ……0

ビットパターンは各カラーが光の三原色のどれを含みどれを含まないかを示している

Figure 5

```
     0 1 1    紫色
AND) 1 0 1    水色
     0 0 1    青色
```

Figure 6

OR，XOR，PRESETの5種類です．PRESETという条件はGETしたグラフィックのパレットコードを反転して（黒↔白，青↔黄，赤↔水，紫↔緑．カラーが反転するとこう変わる）表示するものです．また白黒モードでGETしたデータをPUTするときは，フォアグラウンドカラーとバックグラウンドカラーを指定します．

AND，OR，XORの条件では，指定した位置のパレットコードと，その位置に置く点のパレットコード間の論理演算を行って表示色を決定します．

パレットコードと4種類の論理演算の組合せで，どのような効果が得られるのかを知るためには，数の2進表記と，2進数間で行う論理演算とをよくわかっていなくてはなりません．

まず，パレットコードを表す0から7までの数を2進数に直してみると，*Figure 5*のように3桁のビットパターンが得られます．

パレットコード間の論理演算は，この2進数の桁ごとに行われます（*Figure 6* 参照）．

AND演算	0	1	2	3	4	5	6	7
黒 0	0	0	0	0	0	0	0	0
青 1	0	1	0	1	0	1	0	1
赤 2	0	0	2	2	0	0	2	2
紫 3	0	1	2	3	0	1	2	3
緑 4	0	0	0	0	4	4	4	4
水 5	0	1	0	1	4	5	4	5
黄 6	0	0	2	2	4	4	6	6
白 7	0	1	2	3	4	5	6	7

Figure 7

2進数間で行われるAND，OR，XORのそれぞれの論理演算の詳細は，マニュアルの「論理演算式」の項などを見ていただくとして，ここではパレットコード間で各演算を行った結果だけを表にまとめてみました（*Figure 7〜9* 参照）.

AND演算では2進数の桁に0を持つものが強く，3桁すべて0である黒は最強です．他のどんなパレットコードも黒とのANDをとると黒くなってしまいます．逆にOR演算では2進数の桁に1を持つパレットコードが強く，3桁すべて1である白は，他のどのパレットコードとORをとっても白が出てきます．XOR演算は複雑な結果を示し，見た目には一番おもしろい効果が出せるものといえます

2章以降で，各機能の特性をうまく利用したプログラムをいくつか紹介しますので参考にしてください．

GET命令，PUT命令の両方にある要素ナンバーとは配列変数の何番目の要素にデータを取り込むか，または出力するか決めるものです．これを利用するとひとつのプログラムの中でいくつかの図形をGET，

OR 演算	0	1	2	3	4	5	6	7
黒 0	0	1	2	3	4	5	6	7
青 1	1	1	3	3	5	5	7	7
赤 2	2	3	2	3	6	7	6	7
紫 3	3	3	3	3	7	7	7	7
緑 4	4	5	6	7	4	5	6	7
水 5	5	5	7	7	5	5	7	7
黄 6	6	7	6	7	6	7	6	7
白 7	7	7	7	7	7	7	7	7

Figure 8

XOR演算	0	1	2	3	4	5	6	7
黒 0	0	1	2	3	4	5	6	7
青 1	1	0	3	2	5	4	7	6
赤 2	2	3	0	1	6	7	4	5
紫 3	3	2	1	0	7	6	5	4
緑 4	4	5	6	7	0	1	2	3
水 5	5	4	7	6	1	0	3	2
黄 6	6	7	4	5	2	3	0	1
白 7	7	6	5	4	3	2	1	0

Figure 9

PUT するのにいちいち別の配列変数を使うことなく要素ナンバーだけを指定し直せばよいので便利です．

PAINTについて

PC-8801 mk II は，カラーコードに対応している 8 つの色を使って中間色を作ることができます．どうやって中間色を出すのかというと，ある領域をペイントするとき，その内部の隣接する点をそれぞれ異なる色で塗って，見た目にはそれらの色が混ざっているようにするのです．この役目を果してくれるのが PAINT 命令です．この PAINT 命令には書式が 2 つあります．

```
PAINT (x,y),領域色,境界色
```

この書式は，単に境界色の内部を領域色で塗るためのものです．もうひとつは，さきほどの中間色を表示するための書式で，

```
PAINT (x,y),タイルストリング,境界色
```

となります．タイルストリングは，中間色を出すために各色のドットの並び方を指定するものです．その指定方法はマニュアルに述べられているので，ここでは実際にどんなタイルストリングが何色に見えるのかを *Figure 10* に示しました．この表で，例えば 85,0,0 は，CHR$ (85)＋CHR$ (0)＋CHR$ (0) とすれば，そのまま PAINT 命令のタイルストリングを指定するパラメータとして利用できます．ぜひ利用してみてください．

	85, 0, 0		85, 0, 255
	85, 85, 0		0, 85, 255
	221, 34, 0, 119, 136, 0		0, 0, 119, 0, 0, 221
	221, 0, 0, 119, 0, 0		0, 136, 119, 0, 34, 221
	221, 0, 34, 119, 0, 136		0, 0, 85
	255, 85, 0		0, 0, 34, 0, 0, 136
	85, 85, 85		136, 0, 119, 34, 0, 221
	255, 0, 85		0, 85, 85
	255, 85, 255		0, 34, 34, 0, 136, 136
	85, 255, 0		0, 85, 0
	34, 221, 0, 136, 119, 0		0, 34, 0, 0, 136, 0
	85, 255, 85		0, 221, 34, 0, 119, 136
	255, 255, 85		0, 255, 85
	34, 34, 0, 136, 136, 0		85, 255, 255

Figure 10

キャラクタ表示命令に関する基本的御注意

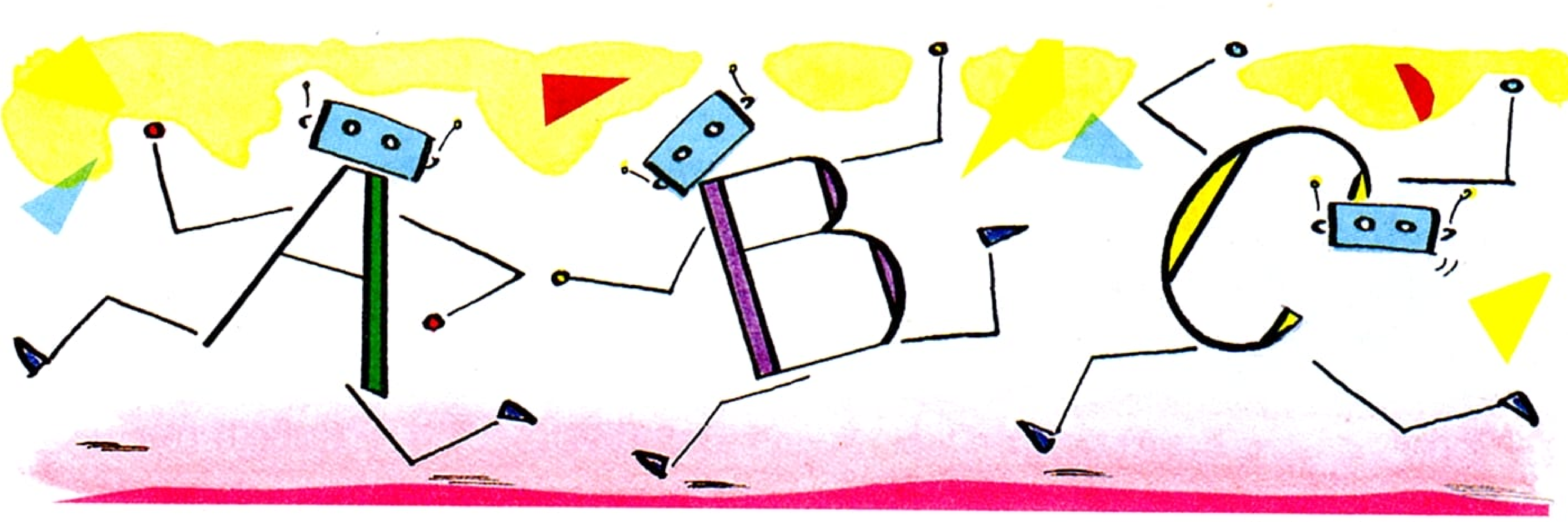
いったいグラフィックの本であるというのに，なぜ筆者はこんな文字表示についての項目の説明をしなきゃあならないんでしょう．グラフィックの話だけしてればいいじゃありませんか．読者の皆さんだってそれを望んでるはずですよね，ぶつぶつぶつ……．と，文句をいっていたら，編集者Ｙ氏がこれを聞きつけました．

――あんたはそういって，グラフィックにばかり肩入れしますがね，わたしゃ何の因果か本の編集なんて仕事をしているもんで，文字というメディアの重要性はよぉく知っているんだ．馬鹿にしちゃいけませんよ．人間が意志を伝える一番の根っこなんですよ．文字ってのは．

ちょ，ちょっと待ってください．いえ何もそう大上段に振りかぶらなくたって，ただほんの少し，メンドクサイナ，なんて本当に頭の片スミでちらっと思っただけ．書きますよ．書けばいいんでしょう．そういえばエジプトの象形文字なんて，あれは大昔のグラフィックですね．あはあは……．

――あんたが作るグラフィックのプログラムだって「リターンキーヲ　オシテクダサイ」なんて表示させる部分があるでしょ，文字が書けなかったらどうするの，「どうしてもリターンキーが押したくてたまらなくなるようなグラフィック」でも考え出してみますかね．

わかりましたよ，もう．コンピュータ・グラフィックスといえども文字の表示をまったく必要としないなんてことはない．だったらやはりそれを美しく画面に出す方法を説明しないのは，片手落ちではありますね．……と，納得した筆者は再びペンを取り，原稿用紙に向かうのでした．

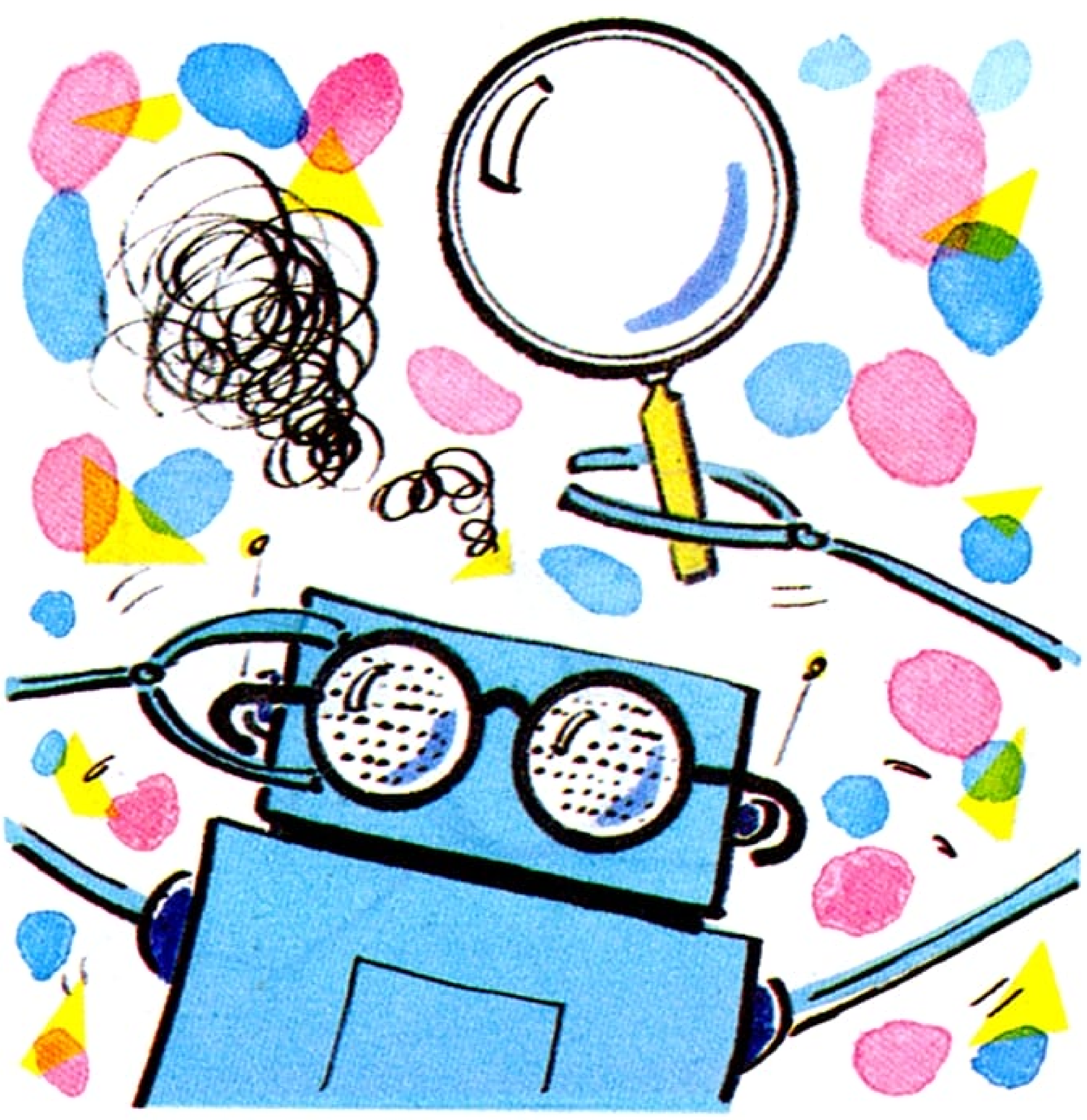


WIDTHと趣味の画面モード

画面の横1行に入る文字数を80字とするか40字とするか，そしてまた1画面の行数を25行とするか20行とするか，それぞれ選択するための命令がこのWIDTHです．

WIDTH命令の存在価値がよくわからん，とおっしゃる方が世の中には2種類いるようでして……．まずは横80字モードに設定した場合の細かい文字がとても読めない，という方．家庭用テレビを接続して画面表示を行っていると，映像がぼやけてしまい，確かに80字モードは少々見づらいようですし，近視の人などは小さな文字を避けたがる傾向があります．

一方，画面の解像力が（テレビよりは格段に）よい専用モニタを使用し，しかも視力はメガネいらずという恵まれた人にとっては，横40字モードというのはむやみにデカイ文字が出てきて邪魔なだけ，こんなモードは必要ないとさえ思えるようです．



まぁ，どちらの派に属する方も，自分の趣味嗜好に合わない文字数モードだからといっても毛嫌いせず，これを必要とする人もいるんだな，と寛容な心を持っていただきたいものです．

筆者らは，どちらかというと「WIDTH 80，25」を実行し，画面一杯に文字を詰めこむのが好きな人間であります．ですから次章以降のプログラムでも，WIDTHの設定を要する場面では，ついついそのような値を採用してしまいました．大きな文字の好きな方々には，この場を借りて衷心よりおわび申し上げます．ドーモスイマセン．

CONSOLEとスクロール

画面の上部にグラフィックを表示し，下の8〜10行ほどを文字表示専用の領域として，種々のメッセージを出力する……．さて，お好きな方にはもうおわかりでしょう，これが今巷でおおいに流行しつつあるアドベンチャーゲームの標準的な画面設定です．

ゲームに限らず，キーボードからのデータインプットを行いつつグラフィック表示をするようなプログラムでは，入力する文字がグラフィックに重ならないようにする工夫が必要です．

それには，まずグラフィックを表示する場所と，データの入出力を行う場所ははっきりと区別しましょう．簡単にいえばグラフィックの上でデータのインプットや，文字の出力を行うなということで，当然といえば当然のことです．PC-8801 mk IIでは，テキスト画面とグラフィック画面がお互い

に独立しているため，グラフィックの上に文字を出力してもグラフィックが消えてしまうことはありませんが，それではやはり美しくないでしょう．

CONSOLE 命令は，グラフィックと文字を同時に使う場合のこのような問題を解決する，最も有効かつ簡単な手段です．この命令はスクロールウィンドウ（文字やデータの入出力を行うための領域）をテキスト画面上の自由な位置に設定します．この書式は次のとおりです．

CONSOLE　開始行，行数，ファンクションキー表示スイッチ，カラー/白黒スイッチ

この実行によって，スクロールウィンドウの，スクロール開始行（画面最上行を 0 として数える）とその開始行からスクロールする行数を指定することができます．画面がスクロールする場合は，このスクロールウィンドウの中だけで行われますし，HOME CLR キーを押しても消去されるのはやはりウィンドウの中だけです．

COLORについて

文字をテキスト画面に出力するとき，COLOR 命令によってその文字の色や機能を指定することができます．

テキスト画面には，カラーモードと白黒モードの 2 つがあり，これは CONSOLE 命令のカラー/白黒スイッチで指定します．COLOR 命令はこの 2 つのモードではその働きが異なります．カラーモードにおいて COLOR 命令で指定するファンクションコードは，0〜7 が順に黒青赤紫緑水黄白のカラーコードに対応していて，出力する文字に色をつけます．ここで 0〜7 の値はパレットコードではないことに注意してください．また白黒モードにおいて 0〜7 の値は，それぞれノーマル，点滅，反転などを表しています．

この COLOR 命令で設定したファンクションコードの値は，それ以後の PRINT 命令で出力する文字に影響します．また，CLS　1，CLS　3 を実行すると初めて，画面全体がその設定に変化します．

では，あらかじめ画面上に出力されている文字について色や機能を設定し直すにはどうしたらよいのでしょうか．ひとつには，COLOR 命令でファンクションコードを設定し直して，それから同じ文字を出力する方法があります．さらに簡単な方法として，COLOR@命令を利用するテクニックがあります．これは対角線で指定する四角形の内部に書かれている文字に，後から色や機能を設定する命令です．

ファンクションコード	カラーコード	文字の表示コード（白黒モード）
0	黒	ノーマル
1	青	シークレット
2	赤	ブリンク
3	紫	シークレット
4	緑	リバース
5	水	リバース，シークレット
6	黄	リバース，ブリンク
7	白	リバース，シークレット

Figure 11

LOCATEによる位置の指定

LINE 命令はその直線の始点と終点を座標で指定することによって，その位置を決定しました．PRINT 命令で文字を出力する場合にも，テキスト画面上でその位置を決めるためにLOCATEという命令が用意されています．

このLOCATE命令で指定できるテキスト画面の座標は，WIDTH命令で宣言した横と縦の行数に制限されています．例えばWIDTH 80，25を指定した後では，LOCATE命令で指定できるx座標は0～79，y座標は0～24までです．これを考慮した上で，LOCATE x，y：PRINT “A”と文字の出力を行えば，テキスト画面上の好みの位置に文字を出力することができます．

ひとつ注意してほしいことは，グラフィック画面上の座標とテキスト画面の座標（キャラクタ座標）の対応についてです．この対応がわからないとグラフィックに説明を入れたりするとき，思うように文字を書き込むことができません．

例えばSCREEN 0で指定したグラフィック画面上に，WIDTH 80，25で指定したテキスト画面を重ねるとします．グラフィック画面は横640ドット，縦200ドットから構成されているので，テキスト画面の1文字は，横が640／80＝8，縦が200／25＝8ドットにあたることがわかります．

従って，グラフィック座標（x，y）で決定される位置におおよそ一致するようなキャラクタ座標は，（x／8，y／8）で表されることになります．

この座標の変換は，グラフィック画面のモードとWIDTHの設定値の組み合わせから，それぞれ計算してみるより他に方法はありません．WIDTH 40，20やWIDTH 80，20などに設定したとき，この対応がどのようになるのか考えてみてください．

WIDTH 80,25の場合

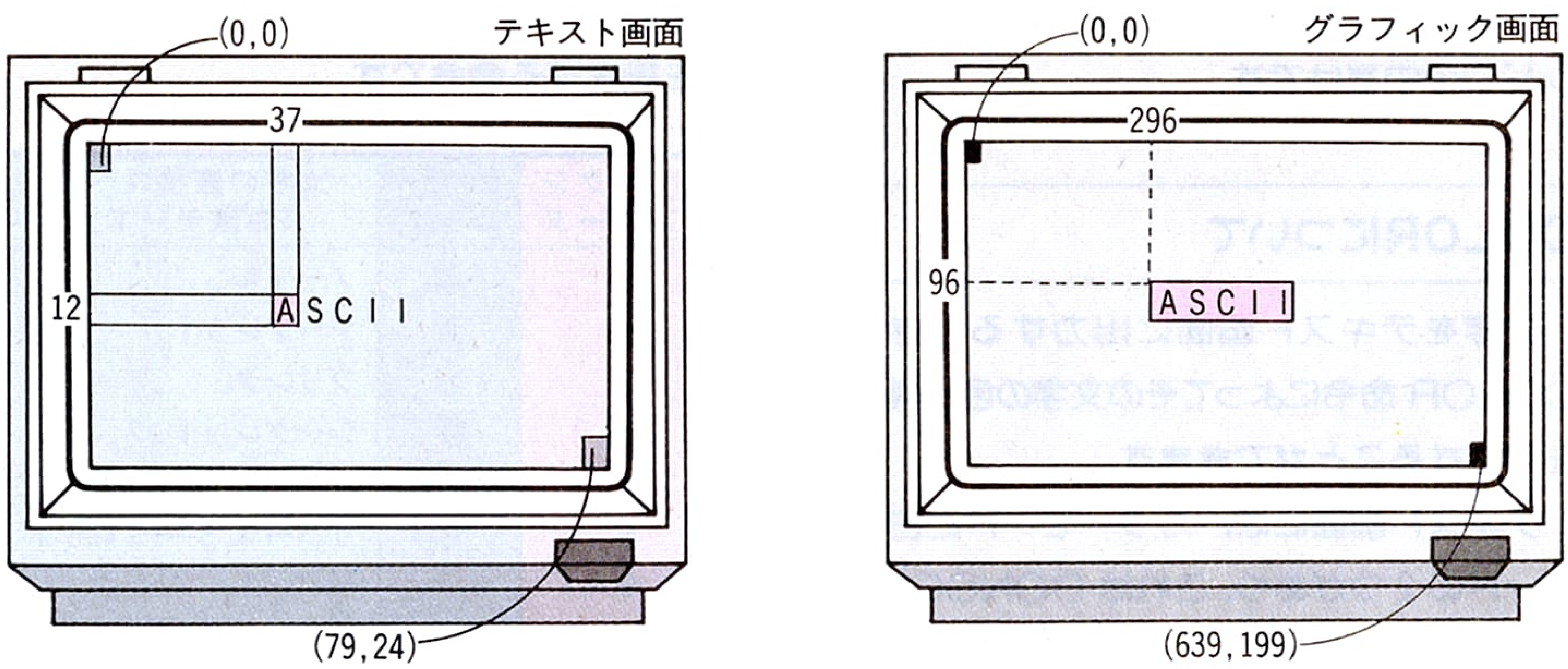
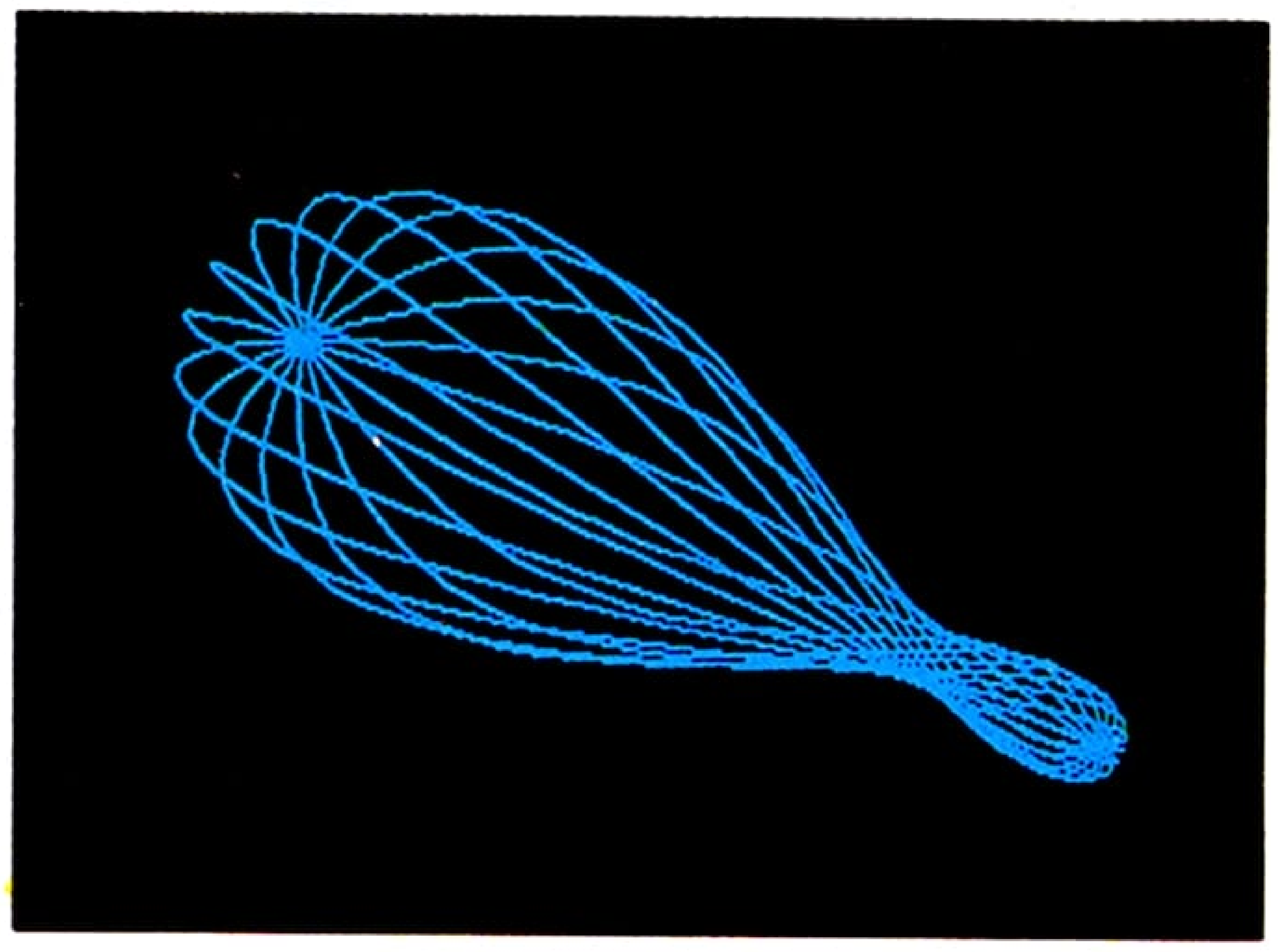


Figure 12

いちじく関数？

座標(X0, Y0)から速度(VX0, VY0)で出発し，座標(X1, Y1)へ速度(VX1, VY1)で到着する物体が，その2点の間でたどる道筋を考えてみます．x座標，y座標ともに媒介変数tの3次関数($0 \leqq t \leqq 1$)として変化すると仮定すれば，簡単な計算で，下記のプログラムの290, 300行に示したような式を得ることができます．ただしこの中のFNA(T)～FND(T)は140～170行で定義している3次関数です（これらの式がどのようにして導き出されたものか私は知っているが，ここにそれを書き記すだけの余地はない．詳細は，フェルマー氏に聞いてほしい！）．



```
100 ' List Ex.1
110 '
120 CONSOLE ,,0 : SCREEN 0,0 : CLS 3
130 '
140 DEF FNA(T)=(T-1)*(T-1)*(T+T+1)
150 DEF FNB(T)=T*T*(3-T-T)
160 DEF FNC(T)=(T-1)*(T-1)*T
170 DEF FND(T)=(T-1)*T*T
180 '
190 C=5 : DEG=3.14159/180
200 FOR I=0 TO 359 STEP 20
210   T0=I*DEG : T1=(I+15)*DEG
220   X0=120 : X1=520 : VX0=900*COS(T0) : VX1=400*COS(T1)
230   Y0=50  : Y1=150 : VY0=405*SIN(T0) : VY1=180*SIN(T1)
240   GOSUB 280
250 NEXT I
260 END
270 '
280 FOR T=0 TO 1 STEP 1/32
290   X=FNA(T)*X0+FNB(T)*X1+FNC(T)*VX0+FND(T)*VX1
300   Y=FNA(T)*Y0+FNB(T)*Y1+FNC(T)*VY0+FND(T)*VY1
310   IF T=0 THEN POINT (X,Y)
320   IF T>0 THEN LINE -(X,Y),C
330 NEXT T
340 RETURN
```

【2章】

グラフィックの基本操作
覚えてほしい，あんなこと，こんなこと

位置を決める

- ●PC-8801 mkIIで用いる座標
- ●中間点を求めて
- ●図形の移動
- ●画面の縦横比
- ●ワールド座標
- ●キャラクタ座標

絵をつくる

- ●コマンドを並べて
- ●READ DATAの利用
- ●角度と距離による位置の指定
- ●乱数の利用

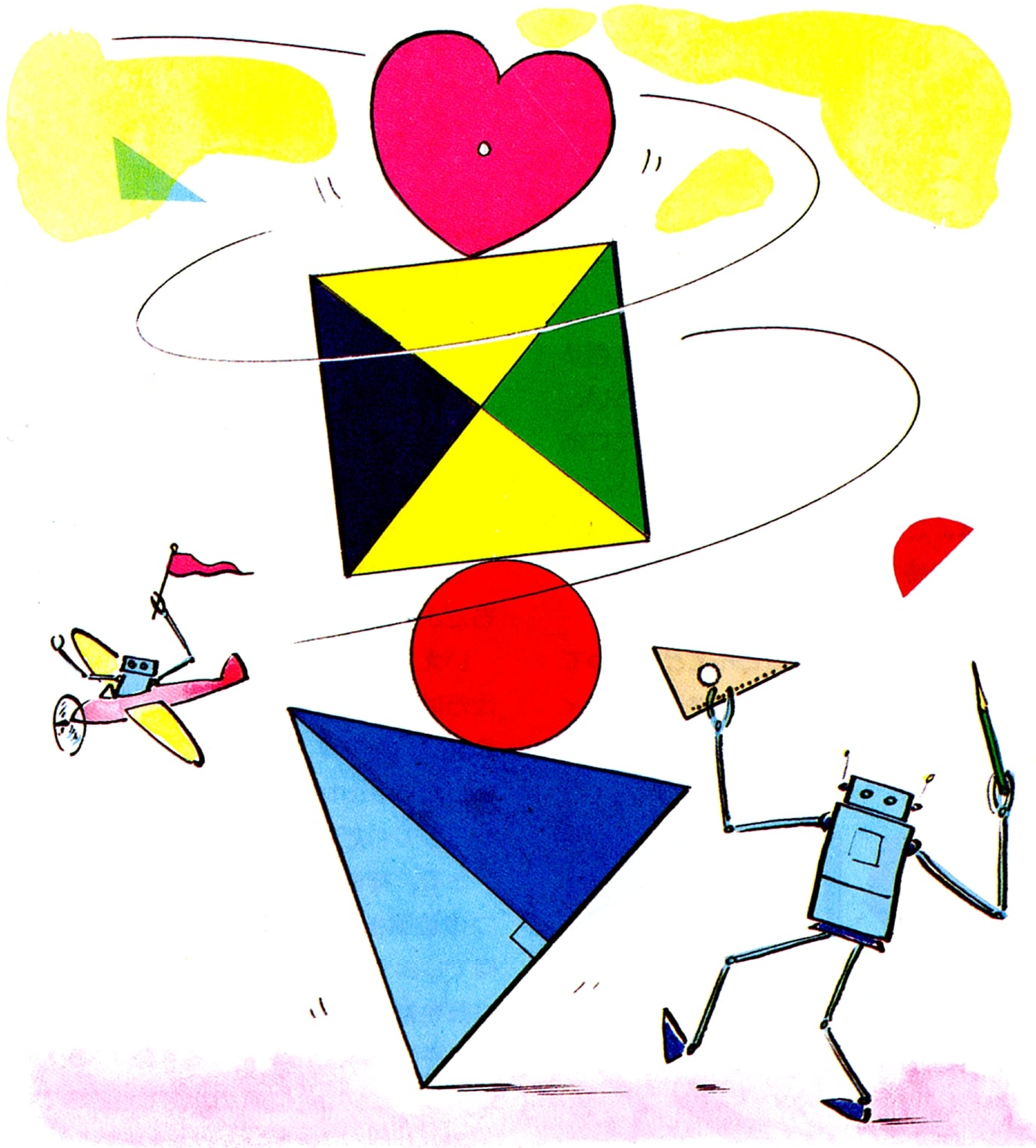
動くグラフィックのこころみ

- ●同じ形を並べて
- ●周期的繰り返し
- ●移動するグラフィック
- ●キー入力による処理
- ●位置の判断
- ●GET & PUT

ここではグラフィックを取り扱うための基本的な方法や考え方についてお話しいたします．

本書はそもそも文章による説明よりもプログラムを主役に据えようという意図のもとに書かれておりまして，何をどうやって計算するとどんなグラフィックが表示できるのかはプログラムを見て読者諸氏に理解していただこう，くだくだしい説明はやめるべし，という半ば教育的，半ばあなたまかせ的主旨が全体を貫いています．手を抜いたのでは決してありません．読者の方々にとってこれが最善の道であると我々は信じているのです．

とはいえ，最初から摩訶複雑なグラフィック・プログラムをポンと皆さんの目の前に放り出し，すごいですねえ，さあ理解してくださいねえ，なんていうのはあまりに無責任すぎるのではなかろうか？ 筆者とし



てその非難は免れ得ません．幾何学に王道はございませぬ，と見得をきった大数学者がいましたが，やはり学問知識というものはすべからく段階を追って基礎から積み上げていくことが肝要なのです．

そこで，本章において基礎的な図形の描き方を説明し，グラフィックのイロハに相当する部分をまず身につけていただこうというのです．ここで述べていることは3章以降で解説していくコンピュータ・グラフィックスの基本となるものばかりです．ぜひとも理解していただきたい！

位置を決める

どこ? ここはどこなの? 少女がそうつぶやいたとき，天の一角から荘厳な声が聞こえてきました．

少女よ，心配するでない．そのまま百歩進み，そこから右手に二百歩進みなさい．おまえの住む世界へ通ずる出口がそこにあるはずだ．

少女は，この声に従って歩いていき，無事に両親の待つ我家へ帰ることができました．めでたし，めでたし…….

えっ! 何をわけのわからんことをいっているのかって? いえ実はこれからグラフィックの一番の基礎となる「位置の指定」についてお話ししようと思いまして．



もうひとつ，これも例ですが，四角い紙の左上あたりに点を打つことを考えてみましょう．

しかもエンピツを持っているのがあなた以外の人物で，あなたは彼女あるいは彼に指示を与えて点を打たせるのだとしたら，その指示は，どんな具合になるのでしょうか．

あっそこじゃない．もっと上．その右，右だよ．あっ行きすぎた…….

いえ，賢いあなたはきっとこんな指示の仕方はしないでしょう．そう，ただこう言えばいいんですから．

上から2cm，左から3cmのところに点を打ってください．

さて，神様の話と四角い紙の話，この2つの位置指定の方法に共通していることは何でしょう．それは，2個の数(データ)で位置を指定している，ということです．そう，平面上のどんな点でも2個1組の数値によってピタリと位置が指定できるのです．

このように位置を指定するための2個1組の数値を座標といいます．コンピュータ・グラフィックスでは，この座標の使い方がまず基礎となります．それではこれから，PC-8801 mk IIでは実際にどんな座標を用いているのか見ていきましょう．

PC-8801 mk II で用いる座標

PC-8801 mk II の SCREEN　0 のグラフィック画面は，おおむね横 8：縦 5 の比率の長方形になっています．横 80 文字モードでは，この中が横方向 640 分割，縦方向 200 分割された，12 万 8000 個（640×200）のドットから画面が構成されています．ドットの形は，*Figure 1* のように縦長の長方形となっています．

これらの中から 1 個のドットを指定するために，PC-8801 mk II では次のような 2 組の数字を組み合わせた座標を用います．

(1) 左端から数えて何個目のドットか

このような横方向の位置を普通 x 座標といいます．

(2) 上端から数えて何個目のドットか

このような縦方向の位置を普通 y 座標といいます．

この x 座標と y 座標の 2 つでドットの位置を指定します．例えば *Figure 2* に示したドットは，左上端から右へ 5 ドット，下へ 2 ドット進んだ位置にあります．そこでこのドットの座標を(5，2)と表示します．

ここで，座標の値は 0 から始まることに注意してください．*Figure 3* のような画面の左上隅にあるドットの座標は，（0，0）なのです．（1，1）ではありません．

ついでにいえば，画面右下隅のドットは，座標（639，199）です．これも間違えやすいのですが，（640，200）ではありません．

x 座標が 0〜639，y 座標が 0〜199，これが PC-8801 mk II の SCREEN　0 のグラフィック画面で指定できる座標の範囲で

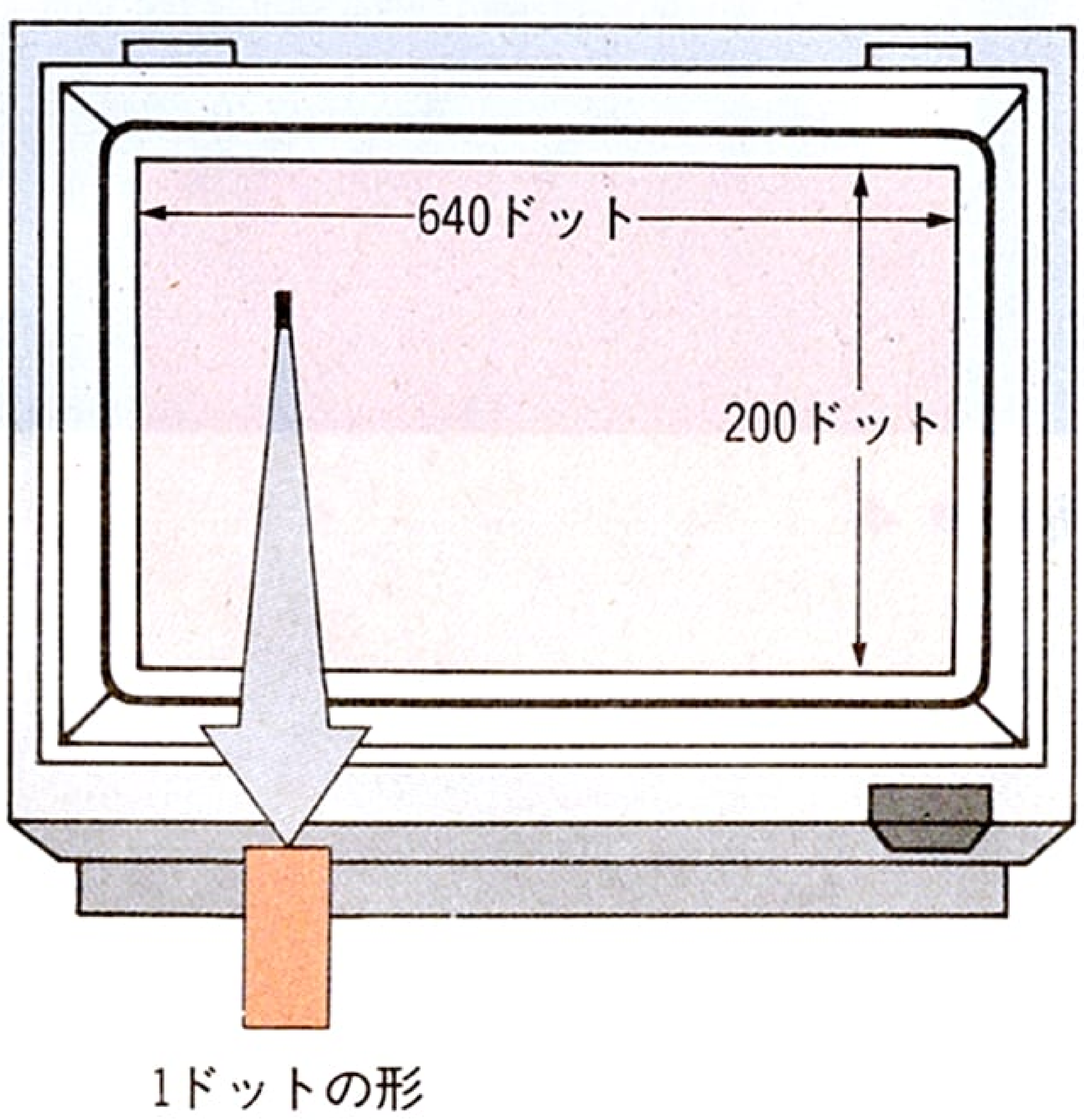


Figure 1

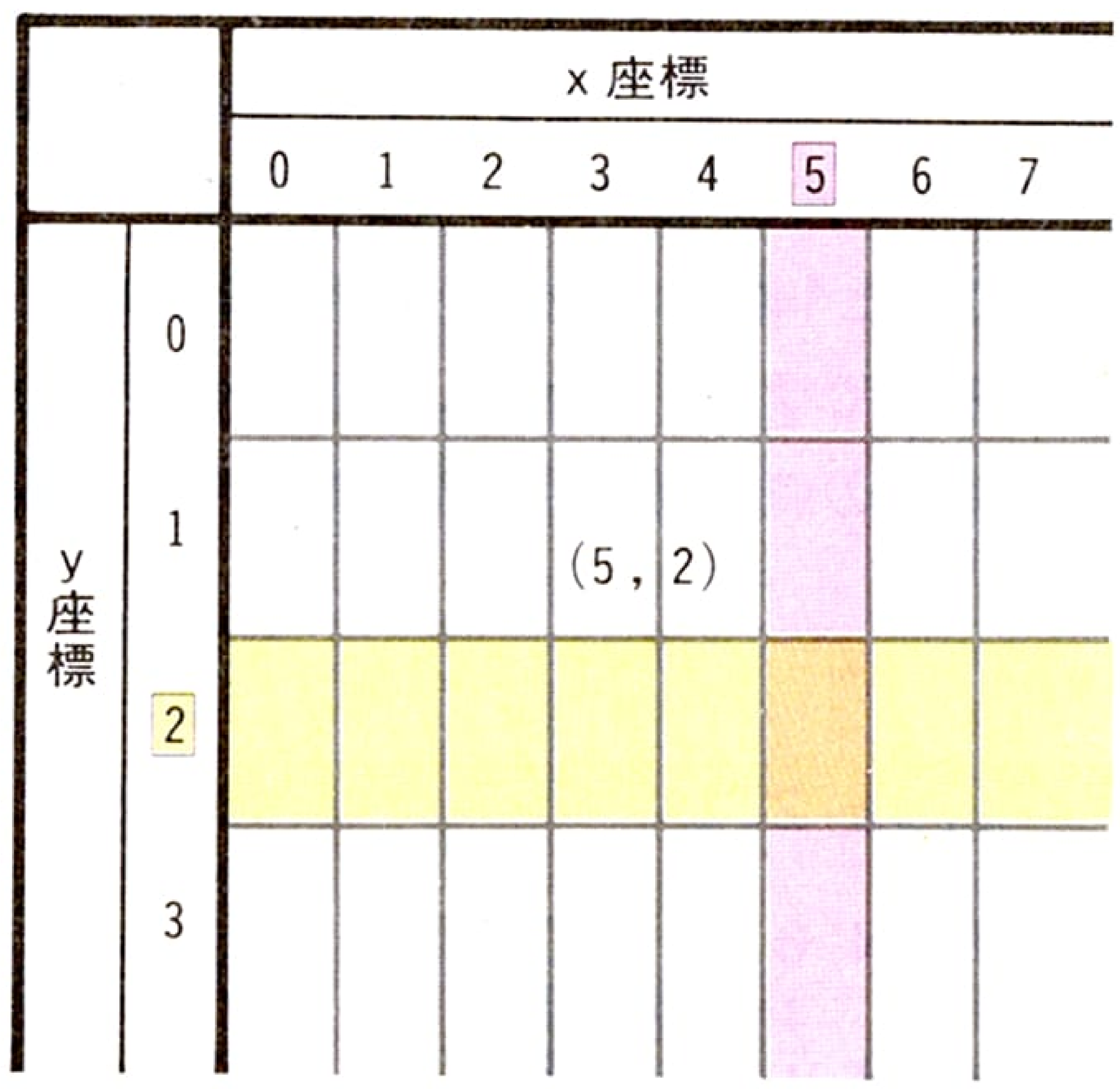


Figure 2

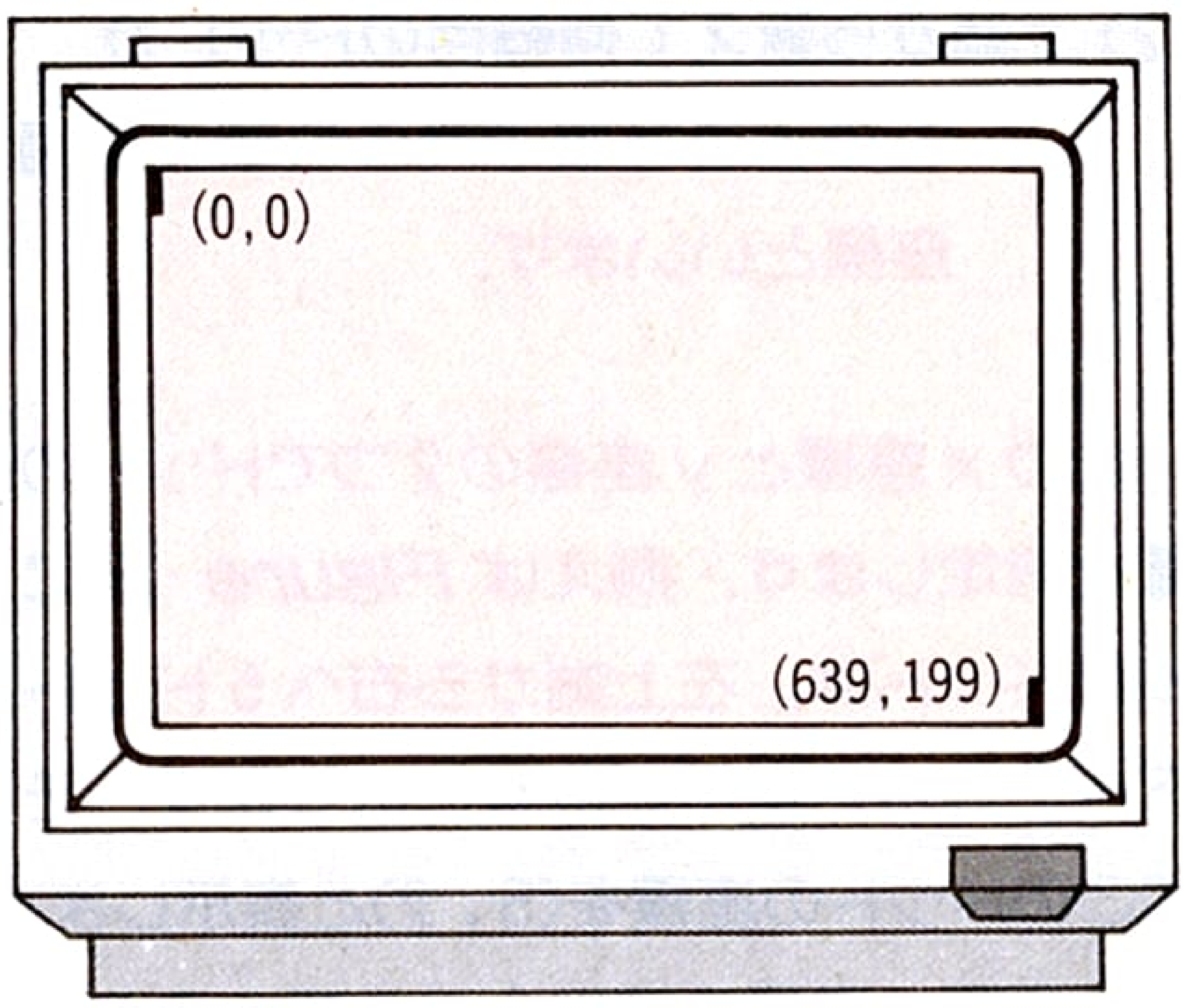


Figure 3

す．どんなグラフィックも，この範囲の中で表現されます．

などというゴタクを並べるよりも，あなた自身の目で確かめるのが一番です．まずは *List 2.1* を打ち込んで実行してみてください．

150 行の意味はわかりますか．これは座標（X 0，Y 0）の点と（X 1，Y 1）の点を結ぶ直線を引いているのです．ここでよく見ていただきたいのは，（X 0，Y 0）という点，そして（X 1，Y 1）という点の位置が画面のどのへんにあるかということです．

130 行と 140 行がこの 2 点の座標を決めている部分ですが，X 0＝20，Y 0＝10 ですから，始めの点の座標は（20，10）で，これは画面の左上あたりで，終わりの点は（620，180），これは右下あたりにあります．それで *Figure 4* のような直線が引かれるのです．よろしいですか？

130 行や 140 行で指定する数値を変えて，画面の位置と座標の関係をつかんでください．例えば，*Figure 4* とは逆に，右上から左下へ線を引くには 2 点にどんな座標を与えればよいでしょうか．横にまっすぐ線を引くには？ 垂直な線は描けますか？ はい，さっそく挑戦してみましょう．

座標の値を変化させるときには，x 座標の範囲が 0〜639，y 座標の範囲が 0〜199 であったことに注意してください．この範囲を超えると何が起きるかって？ それは自分で試してみてください．

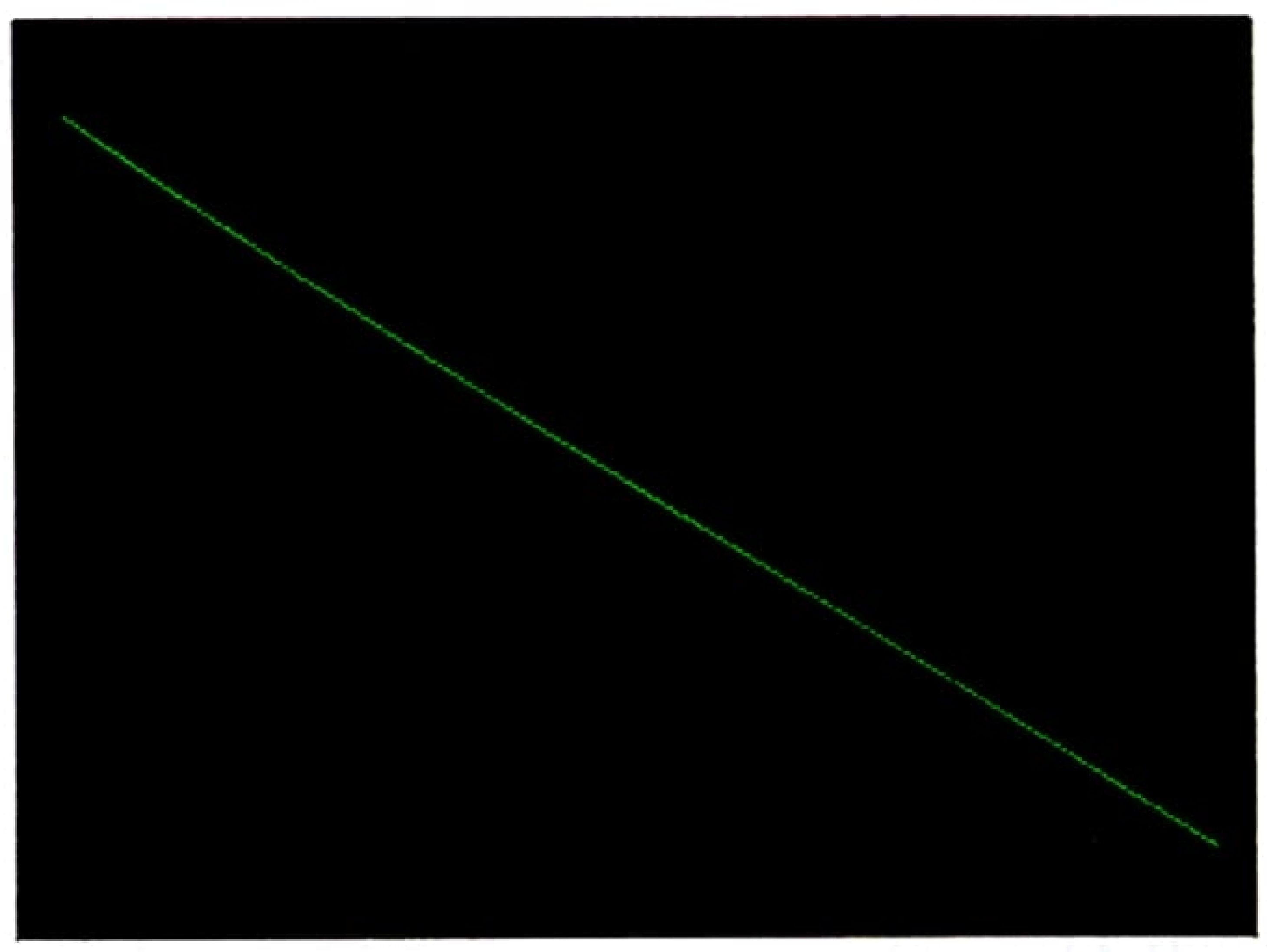


Figure 4

List 2.1

```
100 ' List 2.1
110 '
120 SCREEN 0,0 : CLS 3
130 X0=20  : Y0=10
140 X1=620 : Y1=180
150 LINE (X0,Y0)-(X1,Y1),4
```

中間点を求めて

座標を利用すると，いろいろなおもしろいことができるのですが，その中からいくつか簡単なものを取り上げて紹介していくことにします.最初は「2点の中間の位置を求める」という例から.

ここで,「2点のちょうど真ん中の位置を示しなさい」と言われたら，あなたはどうやってその位置を探すでしょうか．そう，例えば，「日本地図で東京と大阪の中間はどこですか?」と聞かれたりしたら.

ええと，東京と大阪をこうやって直線で結んで,この長さを測ると○ cm.その半分の□ cm の場所は……,わかりました×××でーす（*Figure 5* 参照).

こうやるのも一案ではありますが，ここで座標を使って中間点を計算するたいへんスマートな方法を紹介しましょう.

えっ? 東京や大阪にも「座標」があるのかって!?

はい，立派な座標がございます．それを緯度と経度と申します.

この地球上のどこを探しても，緯度と経度によって指定できない土地はありません.東京しかり，大阪またしかり．地図帳を見てみると，東京は北緯 35.66 度，東経 139.78 度.そして大阪は北緯 34.70 度,東経 135.50 度となっています．ですからこの中間地点は北緯 35.18 度,東経 137.64 度すなわち愛知県と長野県の境に位置する，その名も茶臼山であることがわかります．地図にモノサシを当てる必要も，エンピツで印をつける必要もなく，簡単な計算だけでこの結果が得られたのです.

これはどうやって計算したのかと申しますと，まず緯度は緯度どうし足して 2 で割ります.

$$(35.66+34.70)/2=35.18$$

この値が中間地点の緯度になります．同

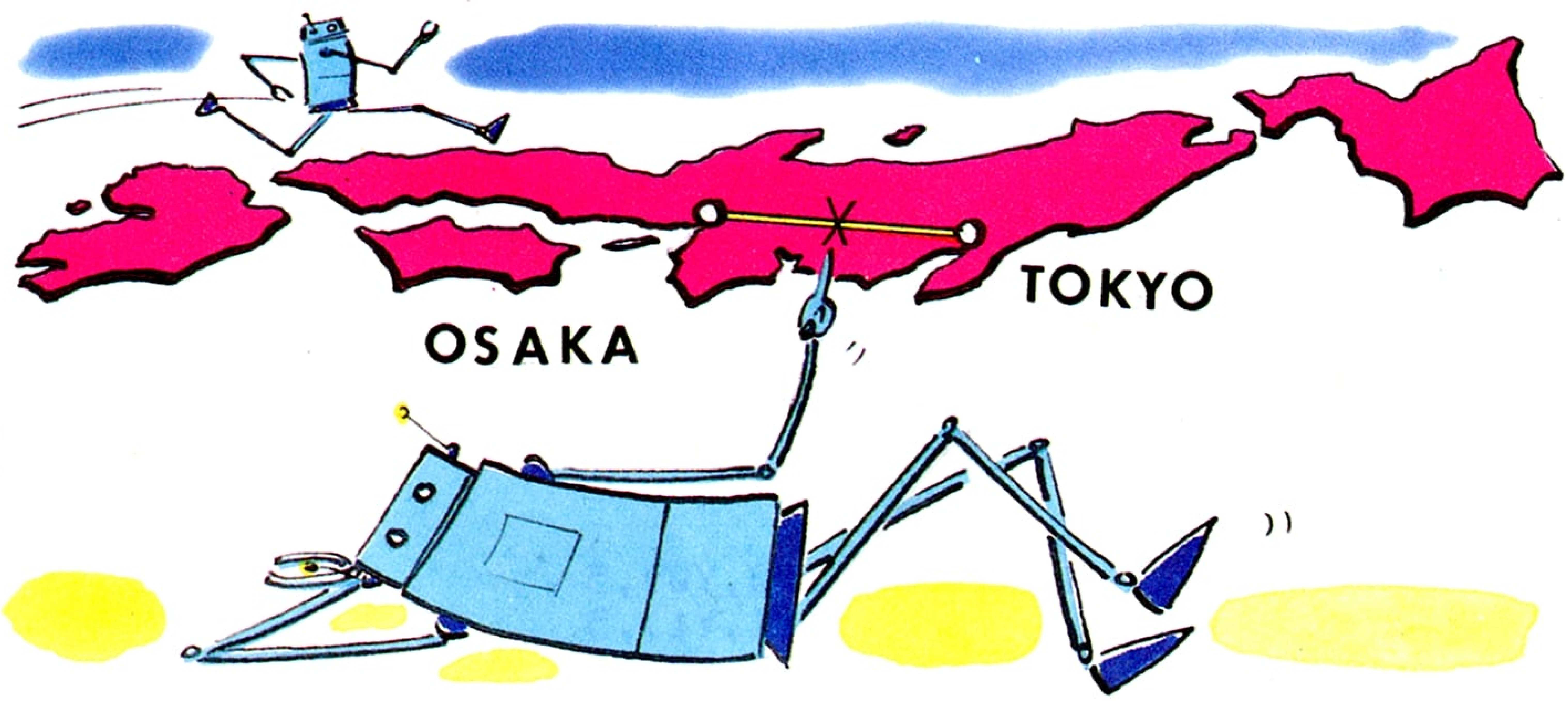


Figure 5

様に経度も経度どうしで計算すると次のようになります．

(139.78＋135.50)／2＝137.64

さて，いよいよコンピュータ・グラフィックスの話に入ります．実は，グラフィックで2点の中間を求めるのもまったく同様に行います．2つの点があったら，そのx座標どうしを足して2で割ったものと，y座標どうしを足して2で割ったものが2点の中間の座標となるのです．

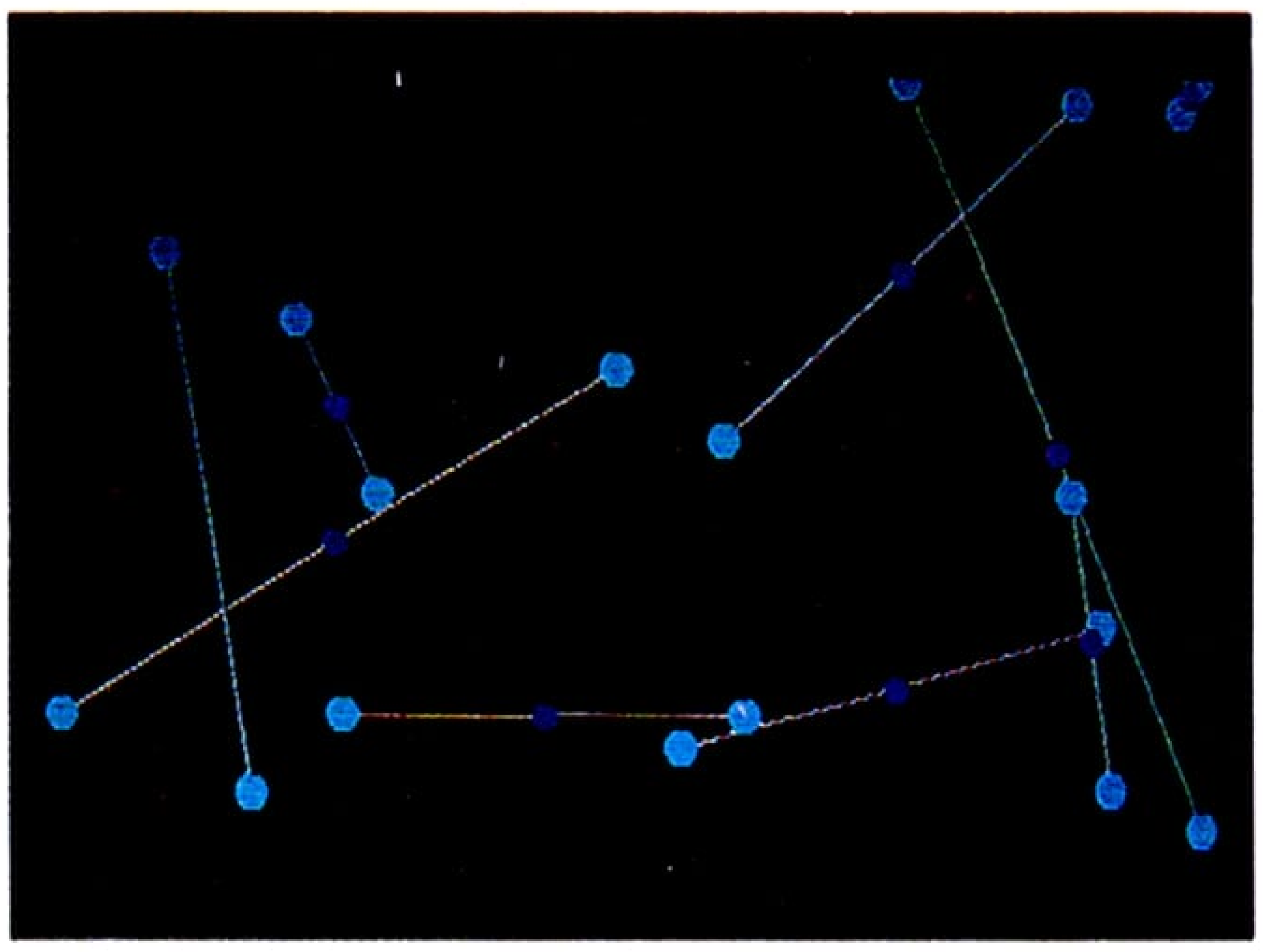


Figure 6

List 2.2 がこれを応用したプログラムです．130，140行で，2点の座標を乱数で決めます．次の150行がお目当ての部分．ここでは，(X0，Y0) という点と (X1，Y1) という点の中間の位置を求めて (X2，Y2) としています．

170〜200行は本当に中間点が求められたのかを目で確かめようというところ．170行で描いた直線の両端と中間点に180〜200行で円が描かれる様子を見てください (*Figure 6* 参照)．

なお，このプログラムは，STOP キーを押して終了してください．

さてここで話を1歩先へ進めましょう．座標を使って2点のちょうど真ん中の位置が計算できることは，これでおわかりいただけたと思います．けれど，いつも中間点ばかりが必要になるとは限りません．ときには端っこから3分の1の点や4分の1の点が欲しい場合もあります．

その場合の計算方法は次のとおり．いいですか，一気にいいますからね．

List 2.2

```
100 ' List 2.2
110 '
120 SCREEN 0,0 : CLS 3
130 X0=INT(RND*640) : Y0=INT(RND*200)
140 X1=INT(RND*640) : Y1=INT(RND*200)
150 X2=(X0+X1)/2    : Y2=(Y0+Y1)/2
160 '
170 LINE (X0,Y0)-(X1,Y1),7
180 CIRCLE (X0,Y0),8,5 : PAINT (X0,Y0),5
190 CIRCLE (X1,Y1),8,5 : PAINT (X1,Y1),5
200 CIRCLE (X2,Y2),6,1 : PAINT (X2,Y2),1
210 GOTO 130
```

※

※ このマークが付いているプログラムは STOP キーで終了してください

点（X 0，Y 0）と点（X 1，Y 1）を結ぶ直線上で，始点（X 0，Y 0）から測って直線の長さのM分のNにあたる点の座標（X 2，Y 2）を求める式は，

```
X2=((M-N)*X0+N*X1)/M
Y2=((M-N)*Y0+N*Y1)/M
```

となります(*Figure 7* 参照)．

これでは何だかよくわからないでしょうから実例を挙げてみます．始点の座標を(20，10)とし，終点の座標を(620，190)とします．そして始点と終点を結ぶ直線上で，始点から5分の2離れた位置を，計算してみましょうか？

この場合は，さきほどの式で，

```
X0=20, Y0=10, X1=620, Y1=190,
M=5, N=2
```

とおいて計算すれば，

```
X2=((5-2)*20+2*620)/5=260
Y2=((5-2)*10+2*190)/5=82
```

つまり，(260，82)であると計算されます．要領はわかりますよね(*Figure 8* 参照)．

この計算式を使ったプログラムが，*List 2.3* です．これは直線を10等分して，端から順に印をつけていきます．180行と190行で直線を10分のNに分ける点の座標を計算し，200行でその位置に円を置きます．これをN=0からN=10まで繰り返すわけです(*Figure 9* 参照)．それでは，N=−1とかN=11とかの値を与えるとどんな結果になると思いますか？

なお，このプログラムは STOP キーで終了します．

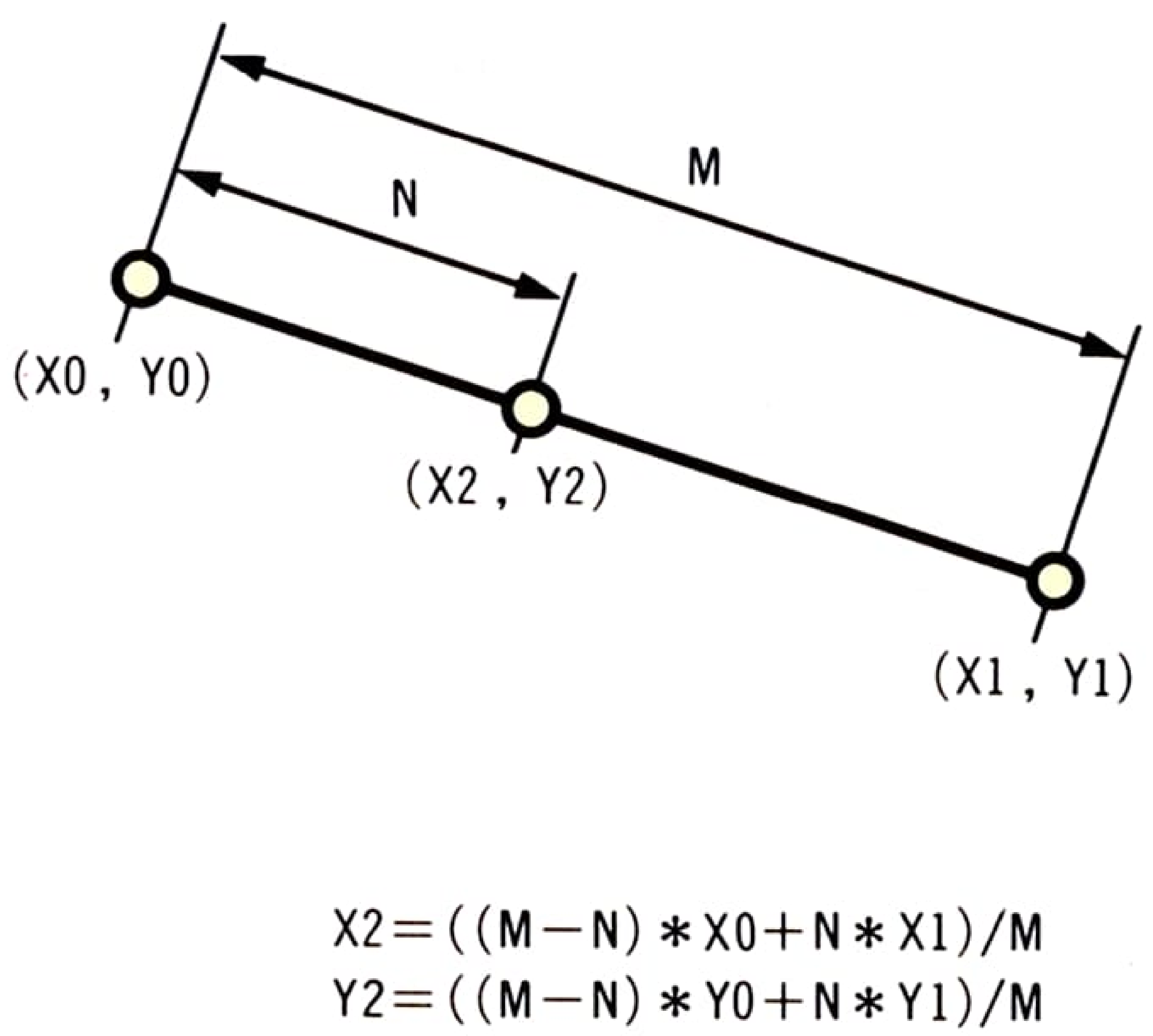


Figure 7

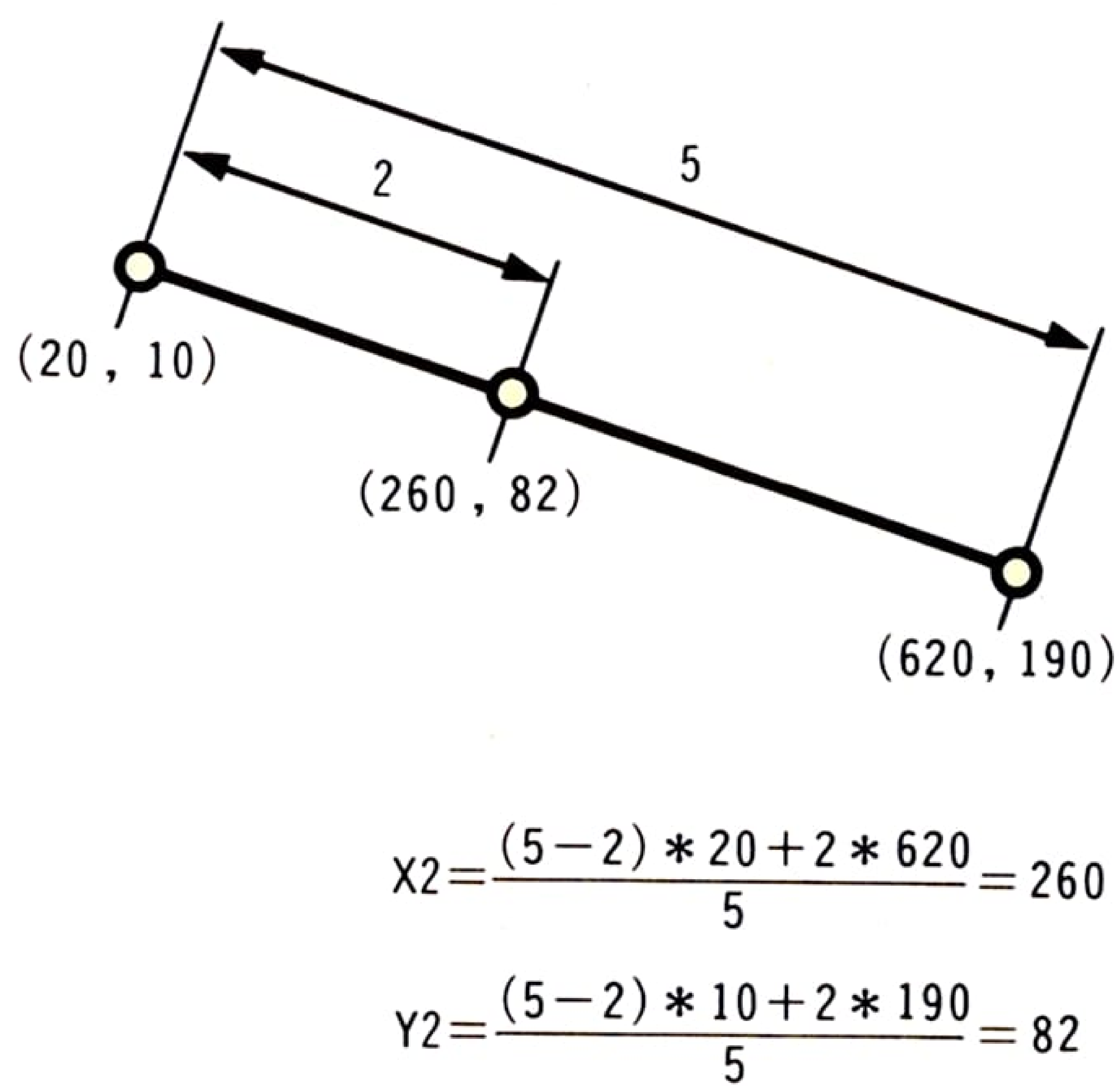


Figure 8

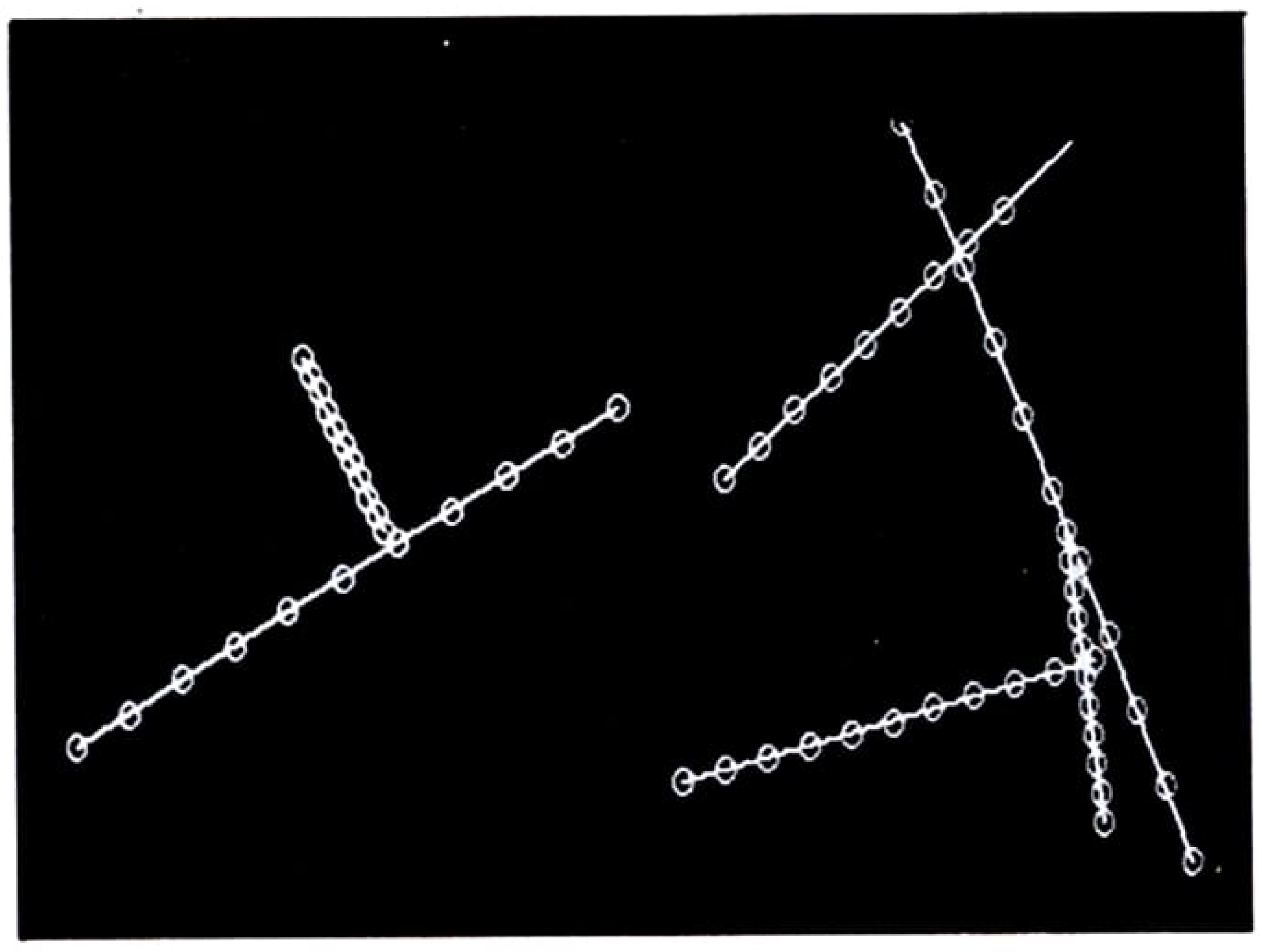


Figure 9

図形の移動

座標を使ってできることの第2弾．図形の位置を動かすための方法を説明します．複雑で華麗なグラフィックももとをたどればここに示すような小さなことの積み重ねでプログラミングされているのです．

まず図形の向きや形を変えないで，位置だけ移す方法．これは図形における各点の座標に一定の値を足す（または引く）だけで行えます．

図形をNドット右へ移すなら，x座標にNを足し，逆に左へ移すならx座標からNを引きます．また，y座標にNを足せば図形は下方へNドット移動し，Nを引けば上方へNドット移動します．

例えば，（X，Y）という座標に描かれた図形を，右へ100ドット，下へ60ドット移動させた場合の，新しい座標は（X+100，Y+60）です．

これを *List 2.4* のプログラムで確かめてみましょう．150行で指定した座標を基本位置として，それを右へ100ドット，下へ60ドット全体的に移動したものが，160行の座標です（*Figure10* 参照）．170行と180行で指定している座標が，150行の座標をどれだけ移動したものかは，もうおわかりですね？

なお，画面に描かれる図形は，150〜180行で座標を指定した後，230行以降のサブルーチンを呼び出して描かれています．このように，同じ図形を1つの画面上に何度も描く場合には，サブルーチン化しておくと便利です．終了するときは，［STOP］キーを押してください．

List 2.3

```
100 ' List 2.3
110 '
120 SCREEN 0,0 : CLS 3
130 X0=INT(RND*640) : Y0=INT(RND*200)
140 X1=INT(RND*640) : Y1=INT(RND*200)
150 LINE (X0,Y0)-(X1,Y1),7
160 '
170 FOR N=0 TO 10
180   X2=((10-N)*X0+N*X1)/10
190   Y2=((10-N)*Y0+N*Y1)/10
200   CIRCLE (X2,Y2),5,3
210 NEXT N
220 GOTO 130
```

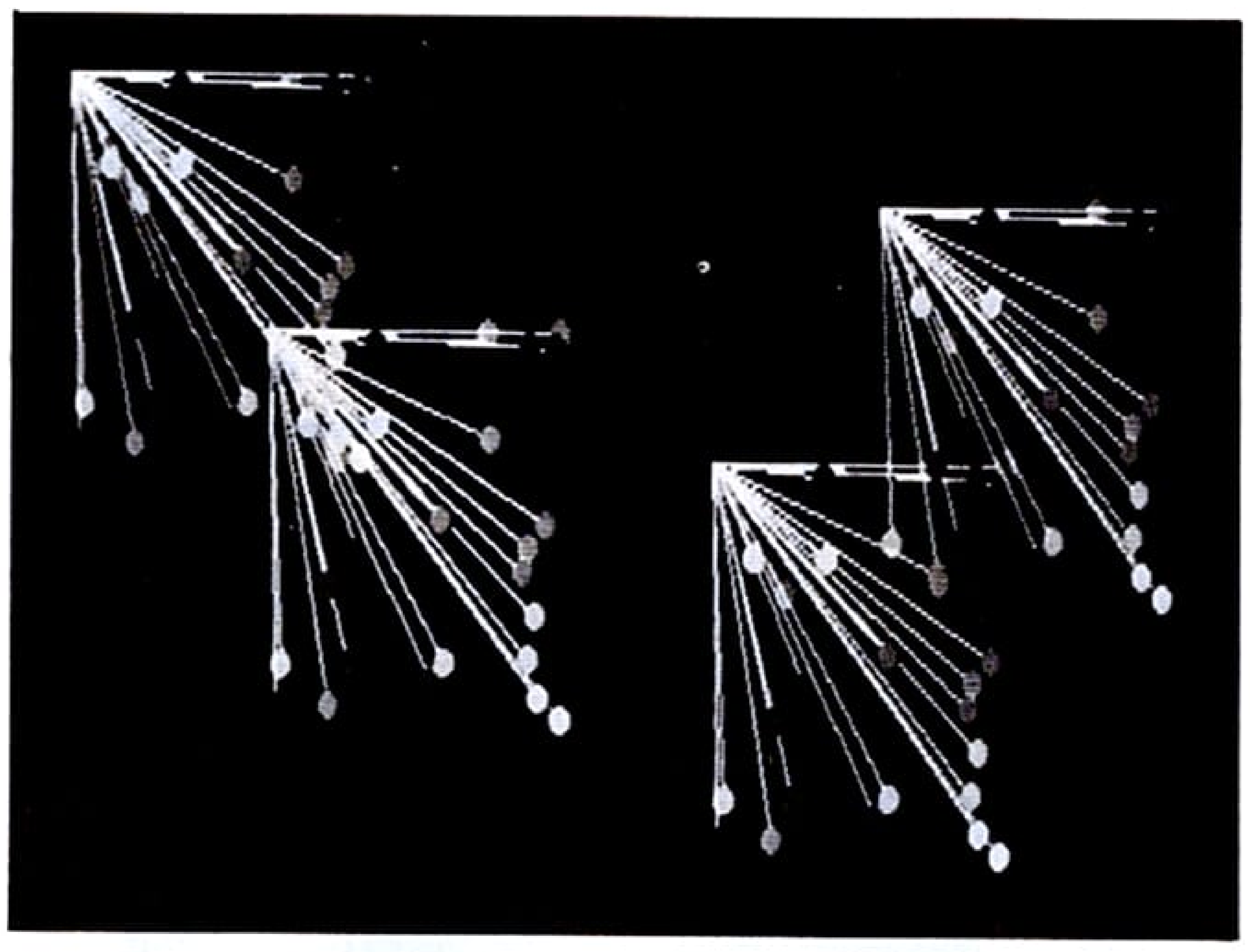


Figure 10

線の終点(X，Y)だけではなく，始点の(0，0)もちゃんと移動させていることに注意してください．図形を構成している全部の点を同時に増加したり，減少したりすることが肝心なのです．

ちなみに，このように向きも大きさも変わらない図形の移動は，「平行移動」と呼ばれております．

こんな「平行移動」は，まだ単純，もっと歯ごたえのあるやつはないのかとおっし

List 2.4

```
100 ' List 2.4
110 '
120 SCREEN 0,0 : CLS 3 : C=1
130 X=INT(RND*150) : Y=INT(RND*100)
140 '
150 X1=0   : Y1=0   : X2=X1+X : Y2=Y1+Y : GOSUB *OUTPUT
160 X1=100 : Y1=60  : X2=X1+X : Y2=Y1+Y : GOSUB *OUTPUT
170 X1=320 : Y1=90  : X2=X1+X : Y2=Y1+Y : GOSUB *OUTPUT
180 X1=400 : Y1=30  : X2=X1+X : Y2=Y1+Y : GOSUB *OUTPUT
190 '
200 C=C+1 : IF C=7 THEN C=1
210 GOTO 130
220 '
230 *OUTPUT
240 LINE (X1,Y1)-(X2,Y2),7
250 CIRCLE (X2,Y2),5,C : PAINT (X2,Y2),C
260 RETURN
```

List 2.4a

```
160 X1=400 : Y1=90  : X2=X1-X : Y2=Y1+Y : GOSUB *OUTPUT
170 X1=639 : Y1=0   : X2=X1-X : Y2=Y1+Y : GOSUB *OUTPUT
180 REM
```

List 2.4b

```
160 X1=0   : Y1=199 : X2=X1+X : Y2=Y1-Y : GOSUB *OUTPUT
170 X1=639 : Y1=199 : X2=X1-X : Y2=Y1-Y : GOSUB *OUTPUT
180 REM
```

ゃる方もいるでしょうね．だんだんむずかしくなっていきますから，ご安心を？

次に説明したいのは，図形を裏返しにする方法，これには左右の逆転と上下の逆転の 2 種類あります．

左右を裏返しにするには，x 座標にマイナス符号をつけます．つまり，(X，Y) という座標に対して (−X，Y) を計算すればよいのです．こうすると，本のページを右から左にパタンとめくるように，図形の左右の向きがひっくり返ります (*Figure 11* 参照)．このとき本の背に当たる位置，つまり移動の中心となるのが，画面でいえば左端，座標でいえば X=0 の地点です．

ただし，ひっくり返した (−X，Y) の位置は，このままでは画面の外へはみ出てしまうので，全体を右方向へ平行移動しなければなりません (*Figure 12* 参照)．それではどれだけ移動させればよいのか．この際，知っていると便利なのは次の法則です．

> 座標 (−X+2a，Y) の位置にある図形は，座標 (X，Y) の図形を X=a の位置を中心として左右裏返しにしたものである

つまり，x 座標にマイナスをつけてさらに右へ 2＊a ドット平行移動させると，それは結果として X=a を中心に左右を逆転した図形になるのです (*Figure 13* 参照)．

これを確かめるために，*List 2.4* の160〜180 行を *List 2.4a* のように変えて実行してみてください．

150 行で指定した基本位置は，160 行で，X=200 の位置を中心として左右ひっくり返されて描かれます．X1 −X というのは −X+X1 と同じことですよ，念のため．

さて次に，上下を裏返すにはどうするか．これは y 座標について，さきほどと同じ操作を行います．つまり，座標 (X，Y) の図

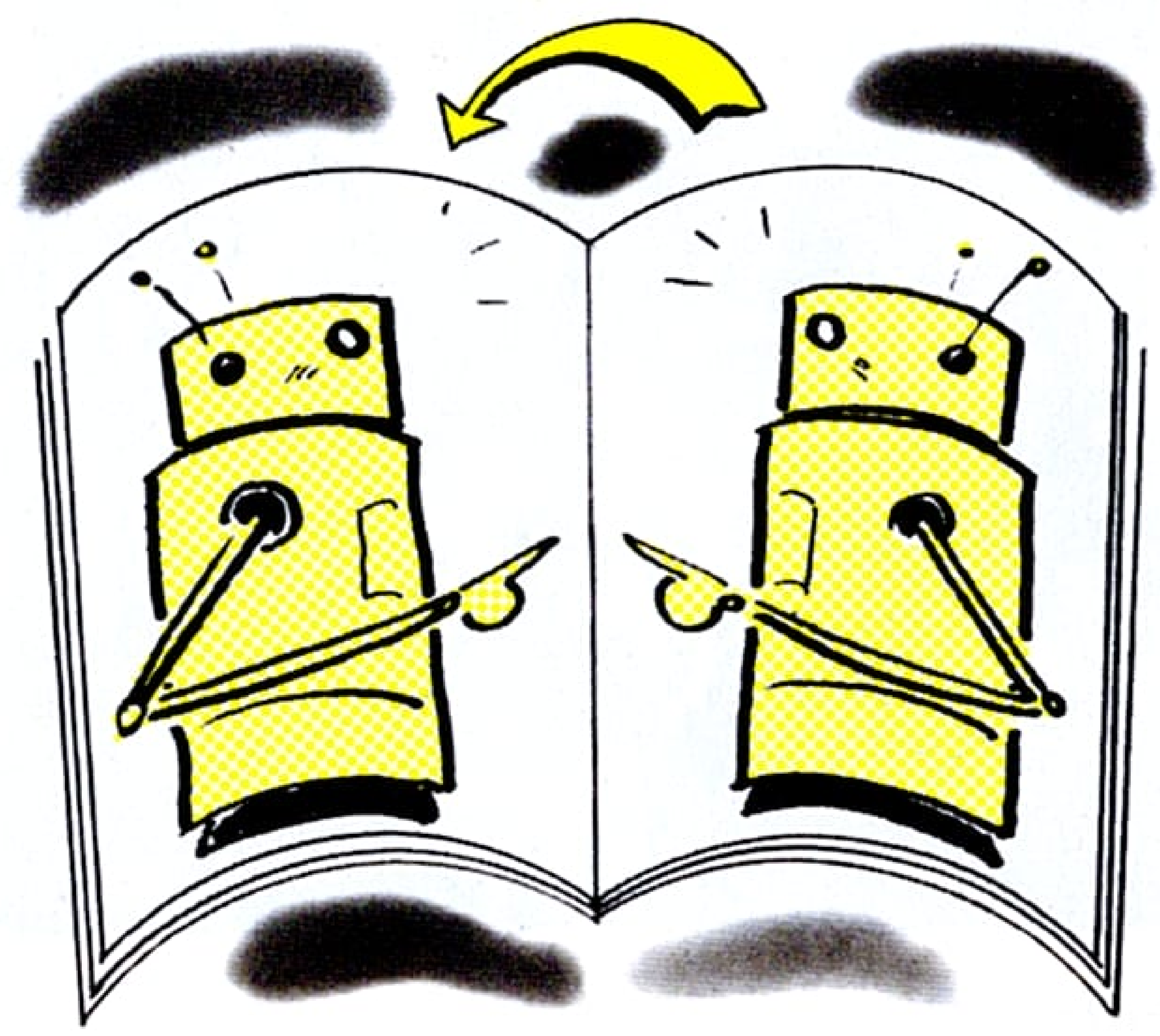


Figure 11

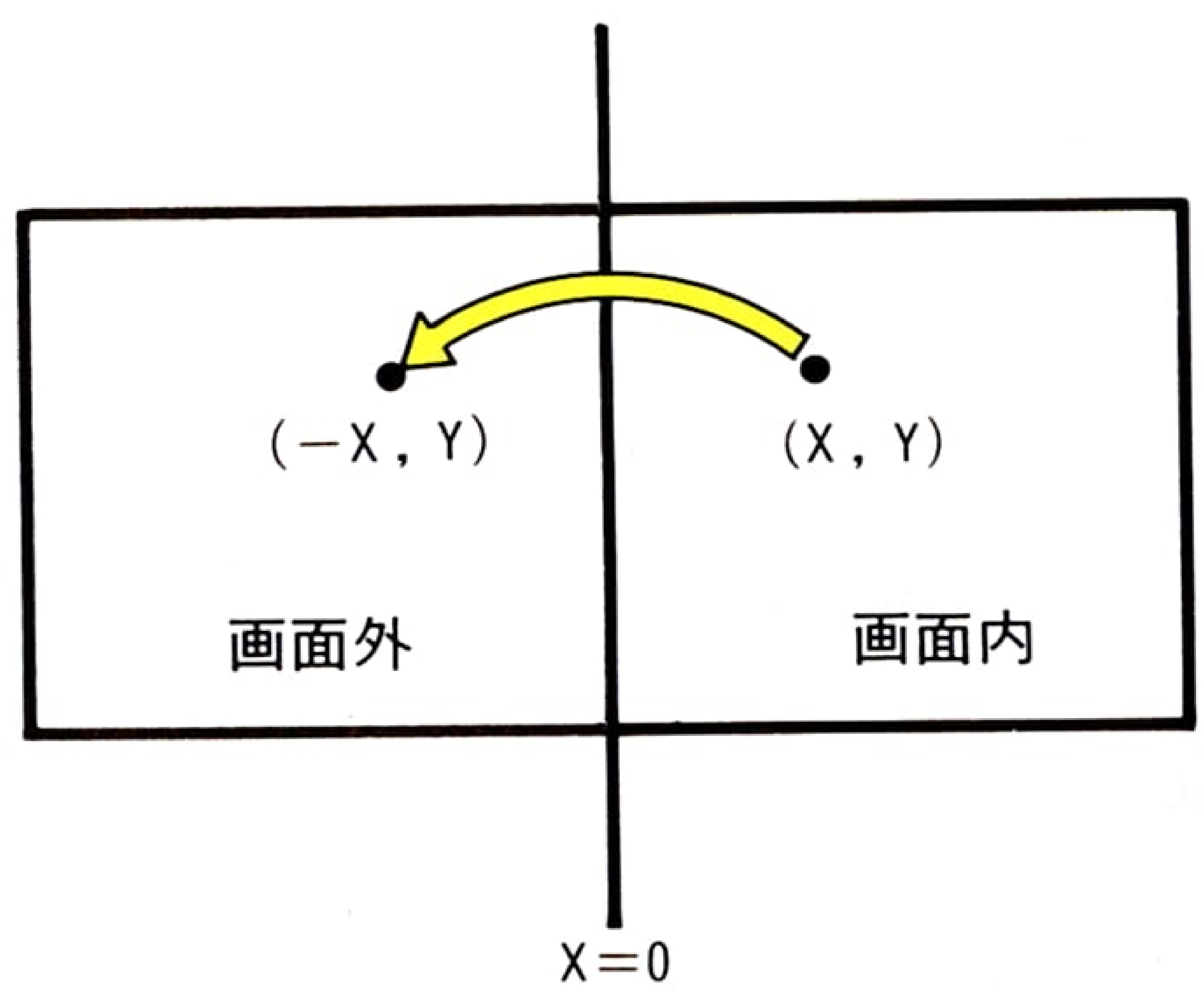


Figure 12

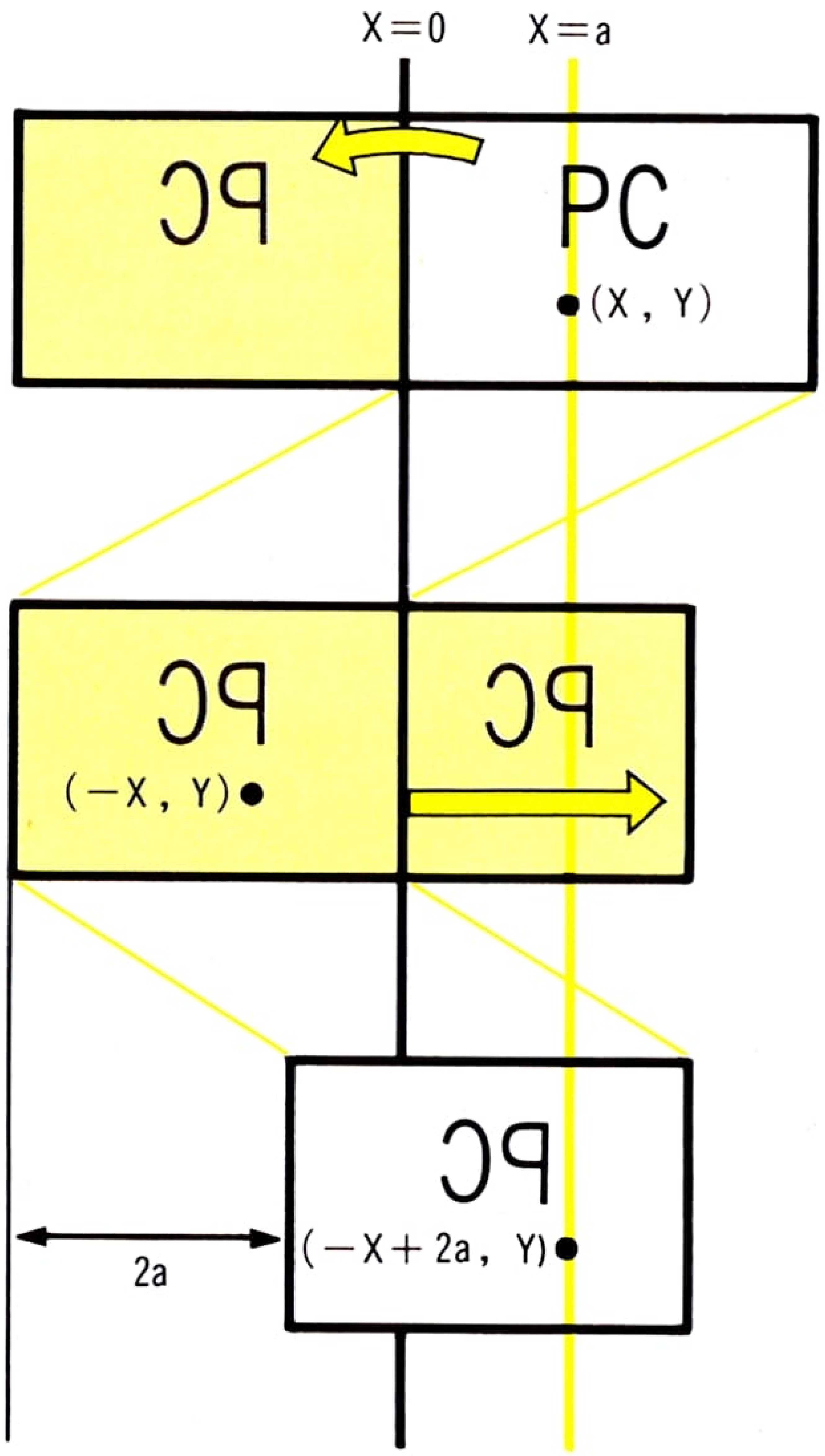


Figure 13

形をY=bの位置を中心として上下を裏返した図形の座標は（X，−Y+2b）として計算できるというわけです．

さらに，*List 2.4* の160〜180行以降を *List 2.4b* のように変えてみてください．

160行が上下の裏返し，そして170行は上下，左右の両方とも裏返しています．

画面の縦横比

横80字モードの1ドットの形は，*Figure 14* のような縦長の長方形をしていますが，

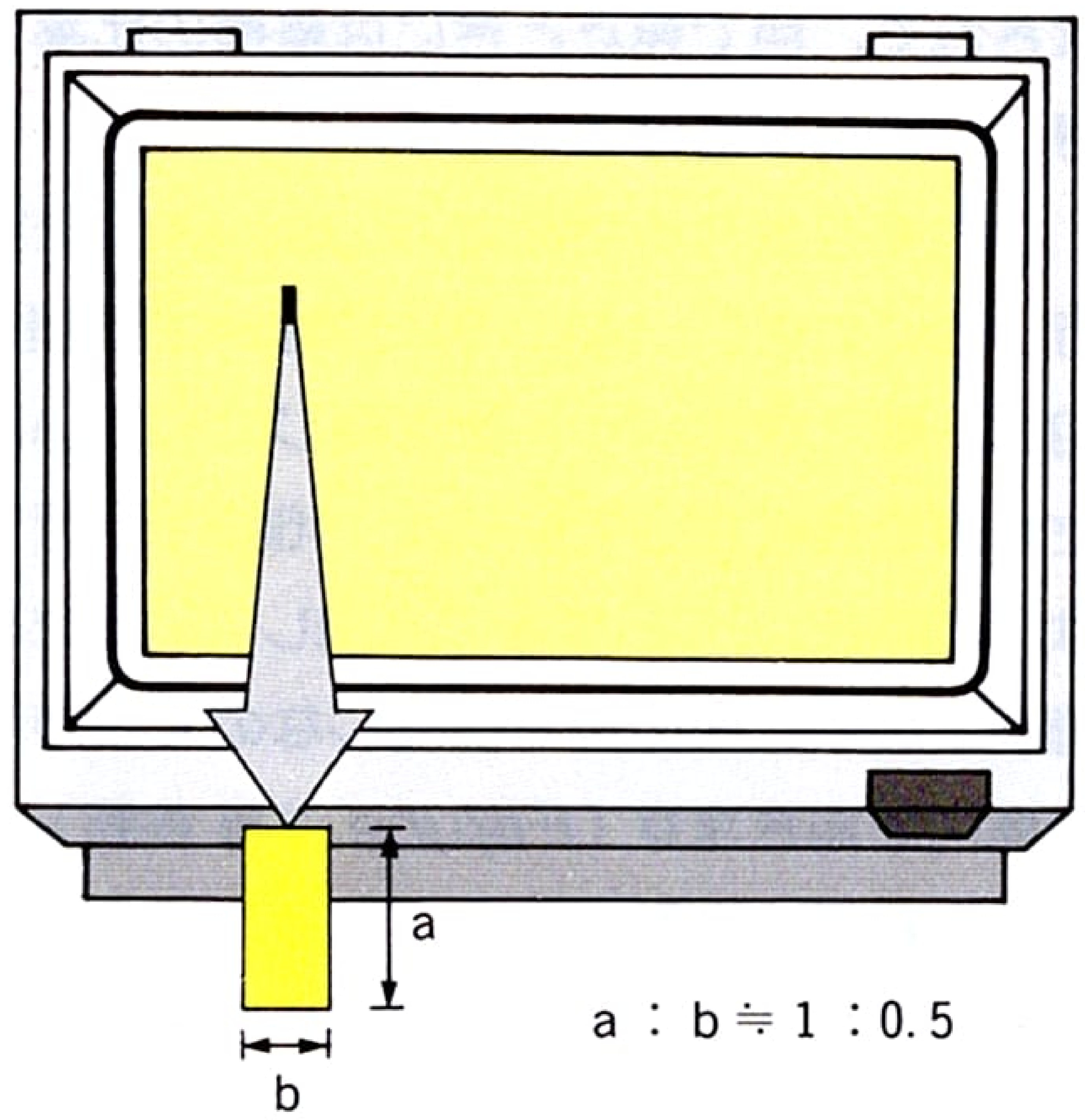


Figure 14

ここでは，そのために生ずる問題を取り上げます．例えば，正方形を描いてください，といわれた場合，あなたならどうしますか．

```
LINE (0,0)-(60,60),7,B
```

このプログラムで正方形が作れるでしょうか？ 実は，このようにして描いた四角形は正方形にはなりません．ドットが縦に長いからです．

きちんとした正方形を描くためには，ドットの縦横比が約1：0.5であるということを考えに入れなければなりません．

横方向に測って，つまりx座標で60ドット分の直線と同じ長さのものを縦方向に描く場合，y座標も60ドットとしたのでは長すぎてしまいます．60＊0.5=30ですから，縦方向には30ドットで十分です．このように，y座標に与える値をあらかじめ0.5倍しておくと，それが画面では縦に長く引きの

ばされて，目で見たときには縦横の比率が等しい図形が描けるのです．

これを確かめるために，*List 2.5* を実行してください．140，150 行で描く四角形，円は，縦と横のドット数を同じにしているため縦に長くなっています．160，170 行では，y 座標のドット数を 0.5 倍しているため画面上の縦と横の長さは同じなので正方形や真円を描きます（*Figure 15* 参照）．

この縦横比に関連したことで，もうひとつ忘れてならないのは 2 点間の距離を求めるという問題．これが意外とひっかかりやすいのです．

2 つの点の距離を計算するときの強い味方は，きっとあなたもご存知でしょうが，三平方の定理またはピタゴラスの定理と呼ばれる数学の定理です．えっもう忘れた？ それでは，簡単に説明しましょう．この定理は，座標(X 1，Y 1)の点と座標(X 2，Y 2)の点の距離は $\sqrt{(X2-X1)^2+(Y2-Y1)^2}$ で求められるというものです．

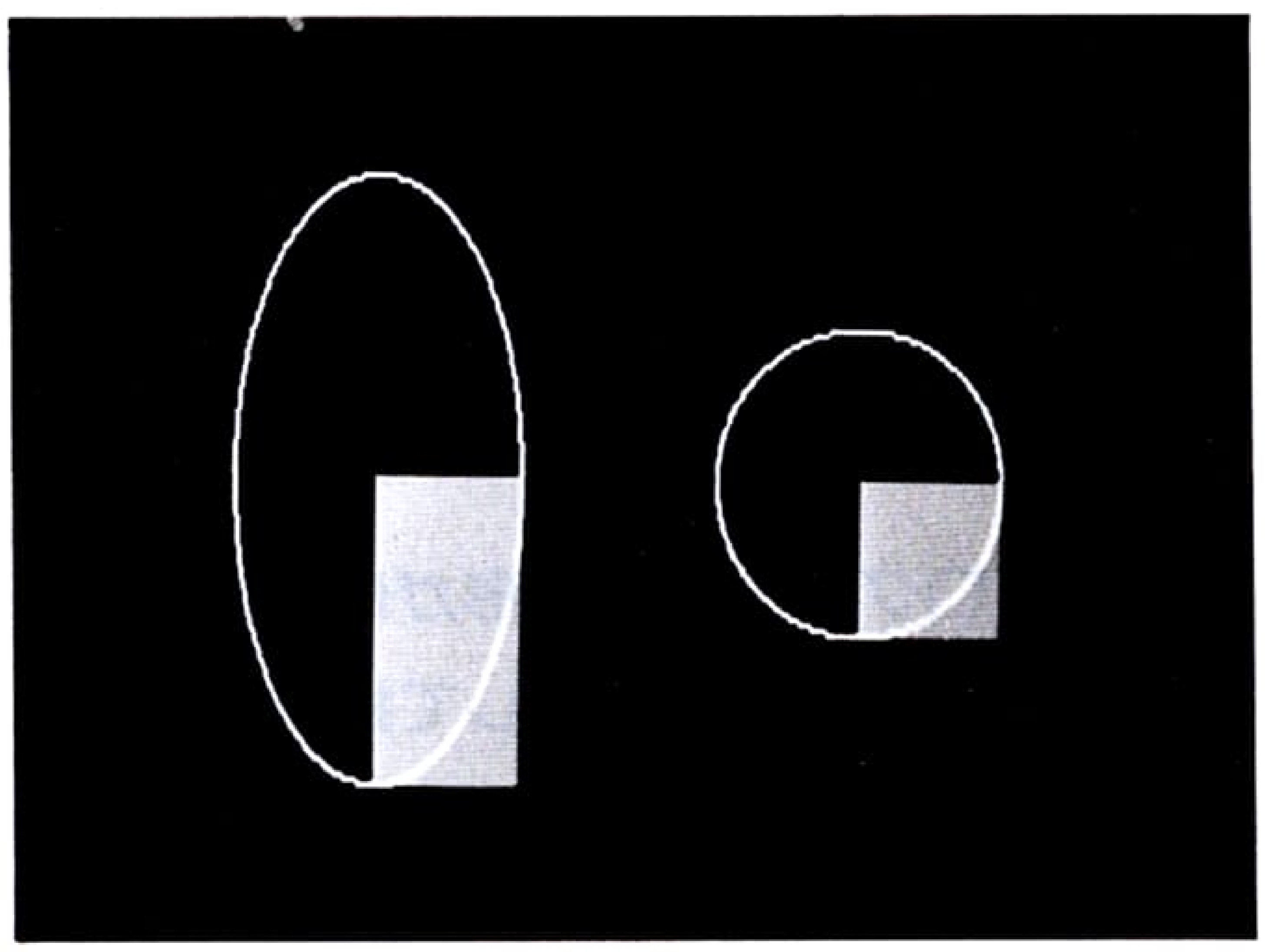


Figure 15

この定理をそのまま利用してグラフィック画面上の 2 点間の距離を求めようとすると……，それは，大きな誤りです．というのもやはり，ドットの縦横比が影響するためです．

例えば，（0，0）の点と（100，80）の点の距離を求める場合を考えてみましょう（*Figure 16* 参照）．

この場合 x 軸方向の幅は 100，y 軸方向の幅は 80 というわけですが，それはあくまでも x 座標，y 座標で数えたドット数のことで，実際の画面上では，さきほどからいっているように y 軸方向は長く伸ばされてしまいます．つまり画面にモノサシを当ててみれば，x 軸方向の 100 に対して y 軸方向の長さは，80 ではなく 80／0.5=160 という長さを持っていることがわかります．ピタ

List 2.5

```
100 ' List 2.5
110 '
120 SCREEN 0,0 : CLS 3
130 P=.5
140 LINE (200,100)-(200+60,100+60),1,BF
150 CIRCLE (200,100),60,7,,,1
160 LINE (400,100)-(400+60,100+60*P),1,BF
170 CIRCLE (400,100),60,7,,,P
```

ゴラスの定理はこの画面上の長さを使って計算しなければなりません．

今までは，y 軸方向の長さを 0.5 倍して y 軸方向のドット数を算出していました．今度は，y 軸方向のドット数がわかっていて，それを画面上の長さに換算するために 0.5 で割るという操作を行ってみます．

List 2.6 を実行してみてください（*Figure 17* 参照）．プログラムは STOP キーで終了します．

150 行でランダムに決めた点が，画面中央に描いた半径 150 ドット（x 軸方向のドット数）の円の内側にあればそこに青い長方形，外側にあれば緑の長方形を描いていきます．これは 160 行で（X，Y）の点と画面の中心点（320，100）との距離を求め，次に 170 行でそれが円の半径 150 より大きいか小さいか判断して色を決めているのです．

160 行で，y 軸方向のドット数の幅 Y−100 を P（=0.5）で割って，画面上での距離になおしてから使っているところがポイントです．

ただし，以上述べてきた縦横比の値は，SCREEN　0 で指定したグラフィック画面

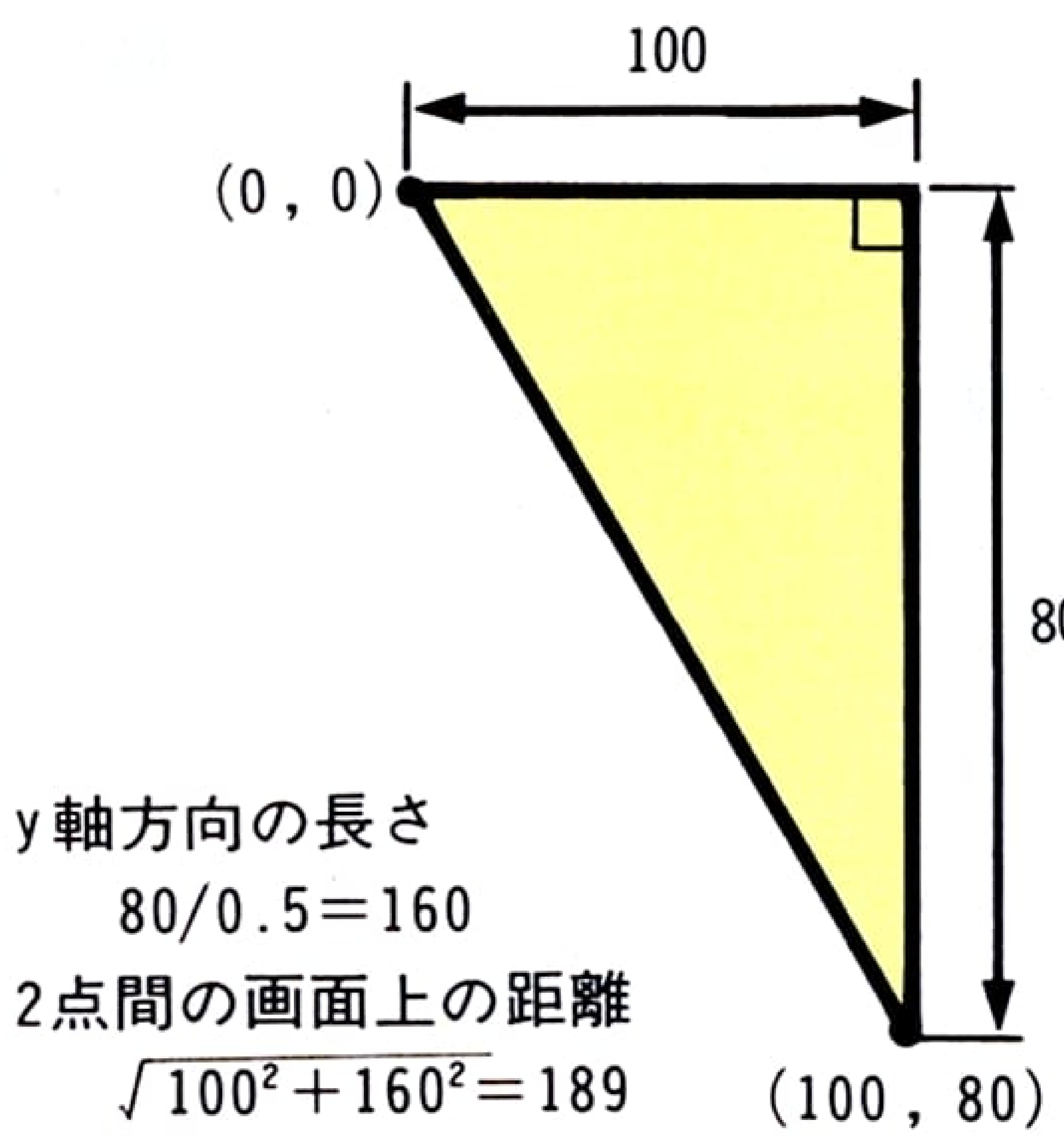


Figure 16

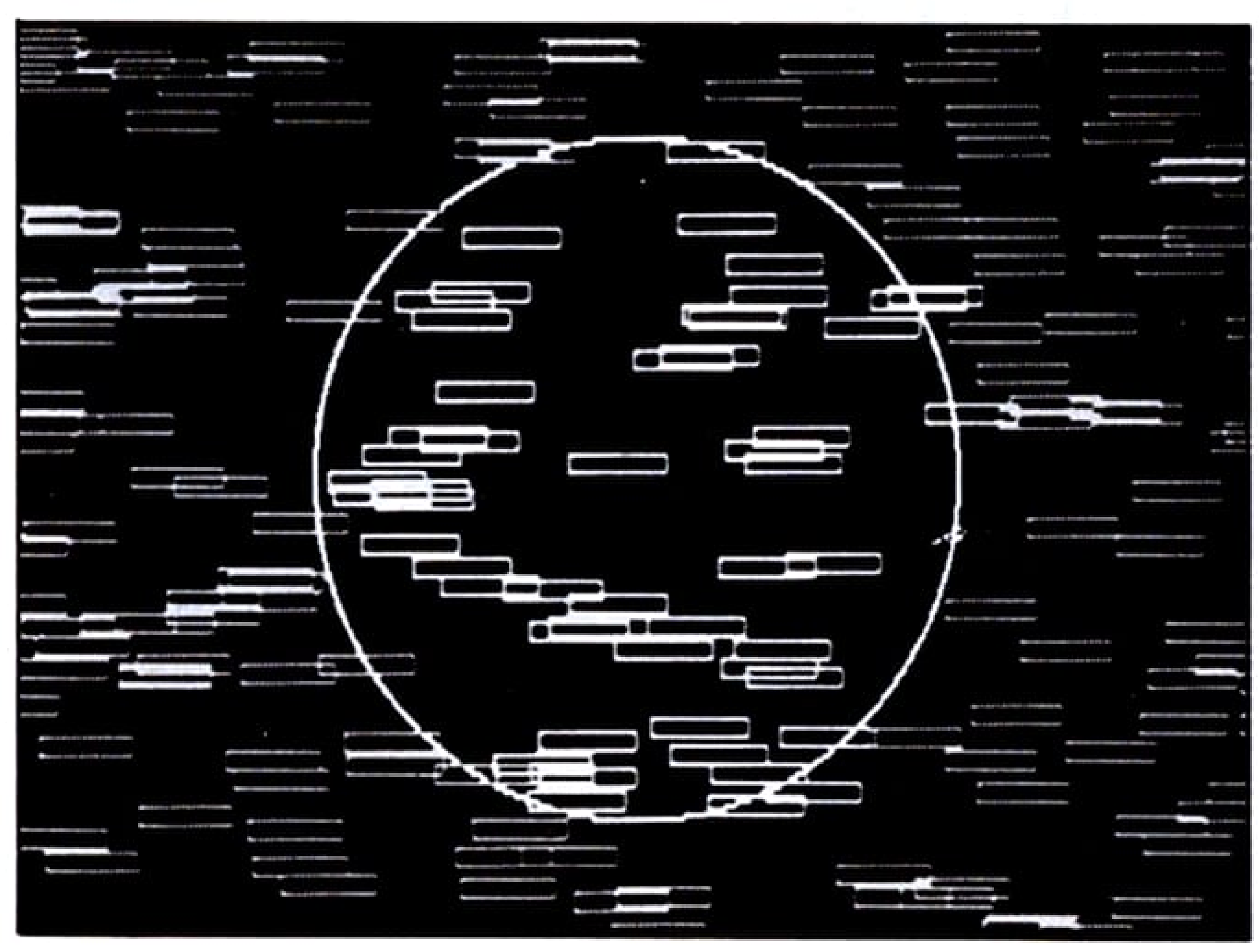


Figure 17

List 2.6

```
100 ' List 2.6
110 '
120 SCREEN 0,0 : CLS 3
130 P=.5
140 CIRCLE (320,100),150,5
150 X=INT(RND*640) : Y=INT(RND*200)
160 R=SQR((X-320)^2+((Y-100)/P)^2)
170 IF R>=150 THEN C=4 ELSE C=1
180 LINE (X,Y)-(X+45,Y+8*.5),C,B
190 GOTO 150
```

における話です．もしあなたが高解像度ディスプレイを持っていて，SCREEN　2(高分解能白黒モード，640×400ドット）を実行したとすれば，その縦横比は1.0になっていることに注意してください．

P.S.　この画面の縦横比を1：1にするために座標に一定の値を掛ける，または，一定の値で割るという手法は，図形の大きさを変えたいときにも用います．具体的な説明は後ほどいたしますが，記憶の片隅に留めておくことを希望する次第であります．

ワールド座標

できあがったコンピュータ・グラフィックスを見て，もう少し大きくしかも縦長にして，右下へ表示したい．なんてとき，あなたならどうしますか？ 大きさを変えるときは座標に一定の値を掛けなければなりませんし，平行移動だったら，座標に何か値を足してやらなければいけません．プログラムの座標データをいちいち全部直すなんて，ほんとめんどうだし，なかなか思うような結果を得ることはありません．じゃ，あきらめるしかないって？ ちょっとまってよ，そんなすぐあきらめてたらグラフィックはできませんよ．ちゃーんと解決策はあるのです．なにかといえば，それはWINDOW命令とVIEW命令なのです．

WINDOW命令とVIEW命令を使いこなすには，座標系ということをしっかり理解する必要があります．もちろん今までだって，画面に位置を決めるには，(x，y)という座標を使ってましたよね．この座標系は，SCREEN　0で指定され，横640，縦200ドットで構成されるグラフィック画面に，そのまま対応しています．つまり，この座標系の範囲はx座標は，0〜639まで，y座標は0〜199までと絶対的に決まってしまっていたわけです．ふーん，そうか．座標系がひとつしかなくってしかもその範囲が決められてるから，移動とか拡大とかがめんどうなのかと思ったあなた．ここで，

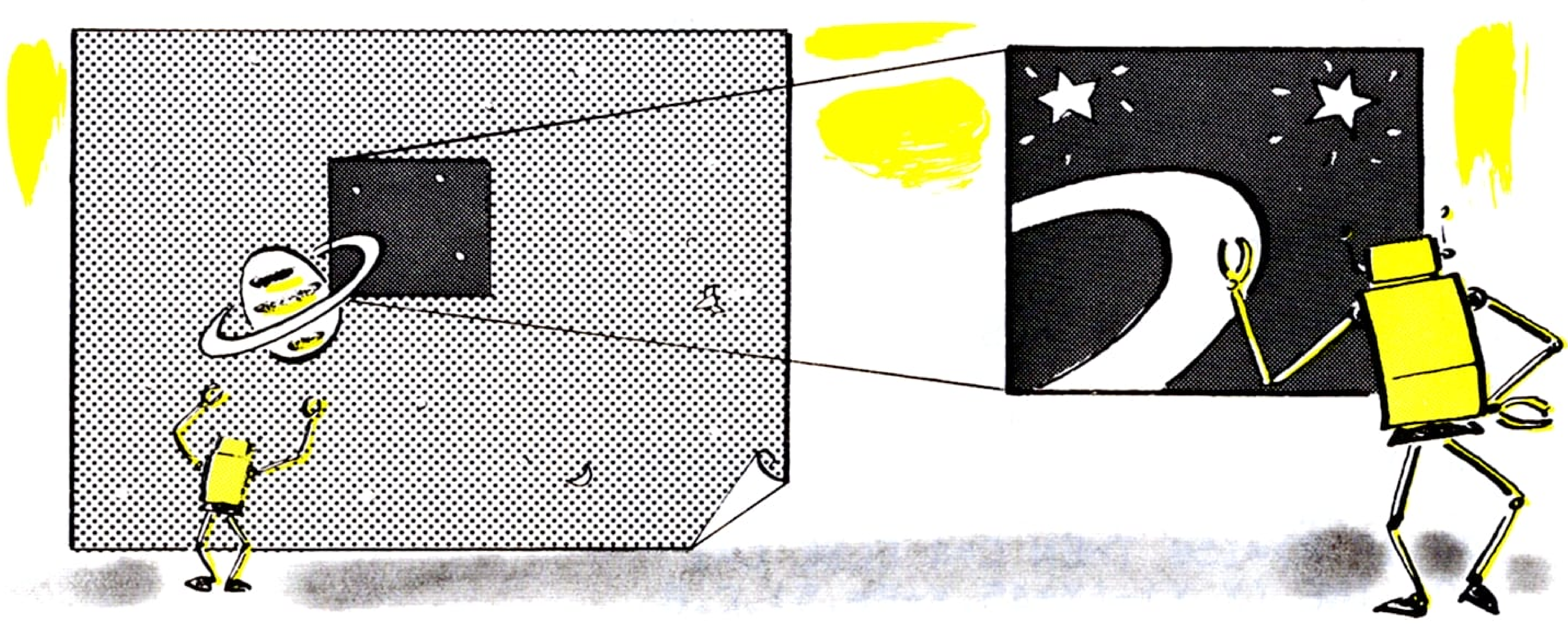


Figure 18

```
LINE (-500,-200)-(1000,500)
```

を実行してみてください.あれっ? おかしいな. x座標もy座標も0～639, 0～199の範囲よりはるかに大きい値なのになんでエラーが出ないの? もしかして, もっと範囲の大きい座標系が他にあるんじゃない?
—— そうです! そこまで考えついたあなたはほとんど正しい. 実は, PC-8801 mkIIのグラフィック座標には, スクリーン座標とワールド座標という2つの座標系があるのです. スクリーン座標は, 先ほどから述べているような, グラフィック画面上に物理的に存在する, 絶対座標のことです. これに対してワールド座標は, 論理座標ともいい, 架空の座標系で, 物理的に存在するものではありません. PSET, PRESET, LINE, CIRCLE, PAINTなどのグラフィック命令は, 座標をワールド座標系で与えます. だから, さっきのLINE命令は, エラーが出なかったのです(*Figure 18* 参照).

さて, 座標系が2つある, というよりもワールド座標があるのはなぜか, もう, うすうすおわかりですよね. LINE命令とかでエラーがでないということもありますが, 最大のメリットは, 図形の移動つまり変換が楽だってことです. 図形を移動したり, 大きさを変えたり, 縦横比を変えたりすることを変換というのはご存知だと思います. その変換を行う際, スクリーン座標しかなければ, その座標系の上で, 座標をあちこちいじらなければなりません. ところが, ワールド座標系に, もととなる図形を描いておいて, その図形をそのままスクリーン座標上にもってくれば, とても簡単にいろいろな変換を行うことができるでしょう.

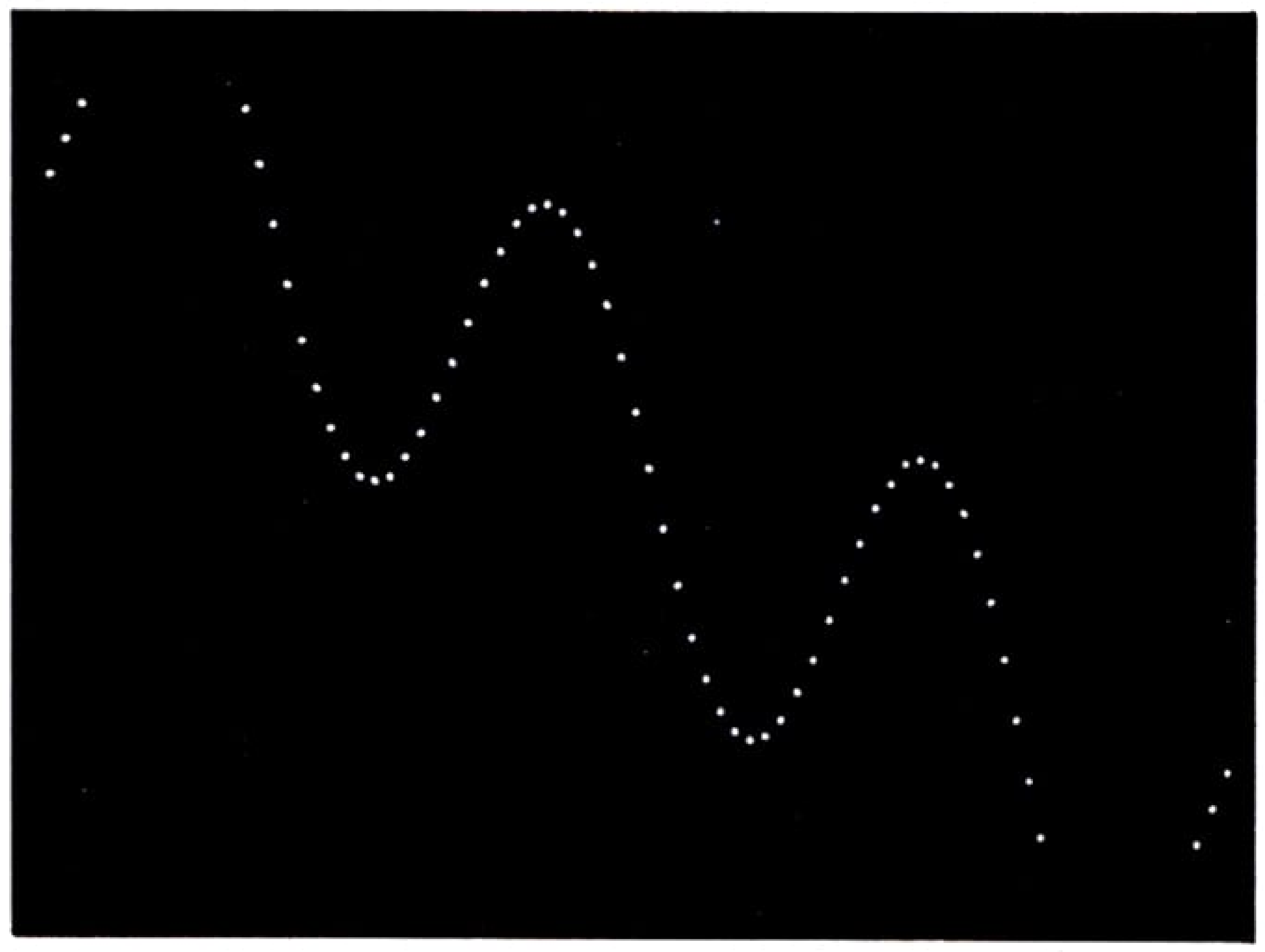


Figure 19

スクリーン座標とワールド座標の話, わかっていただけましたか? *List 2.7a* と *List 2.7b* は, WINDOW命令を使ったプログラムとそうでないプログラムの例です. どちらも *Figure 19* のような結果になります.

さて, WINDOW命令とVIEW命令の説明をしましょう. WINDOW命令とVIEW命令は, 非常に大きな範囲をもつワールド座標系(論理座標)に描かれた図形の必要な部分を, 絶対座標の好きな所へ, 好きな大きさで表示する(変換する)という命令です. WINDOW命令の書式は次のとおりです.

WINDOW $(x_1,y_1)-(x_2,y_2)$

- (x_1,y_1): 表示したい図形を含む範囲の始点
- (x_2,y_2): 表示したい図形を含む範囲の終点

ただし, $x_1<x_2$, $y_1<y_2$

WINDOW 命令は，ワールド座標のどの部分を画面に対応させるかを決めています．グラフィックを描く場合は，WINDOW 命令と VIEW 命令を両方指定しないと，図形の変換はできません．VIEW 命令は，WINDOW 命令で指定した範囲を画面上のどこに表示するかを指定するものです．WINDOW 命令と VIEW 命令の指定に使われる座標系が異なることはもうおわかりですね．WINDOW 命令がワールド座標系，VIEW 命令が，スクリーン座標系です．VIEW 命令の書式は次のとおりです．

VIEW $(x_1, y_1)-(x_2, y_2)$

- (x_1, y_1)：画面上の，図形を出力したい範囲の始点
- (x_2, y_2)：画面上の，図形を出力したい範囲の終点

ただし，$0 \leqq x_1 < x_2 \leqq 639$，$0 \leqq y_1 < y_2 \leqq 199$

WINDOW 命令と VIEW 命令のパラメータを変更すれば，図形の移動，図形の比率の変更などの変換ができます．例えば，*List 2.7b* に，145 VIEW（100，50）ー（500，150）という行をつけ加えれば図形が縮小されます．

List 2.7a

```
100 'List 2.7a
110 '
120 SCREEN 0,0 : CLS 3
130 DEF FNF(X)=X+SIN(5*X)
140 '
150 FOR I=-2 TO 2 STEP .05
160   X=639/4*(I+2)
170   Y=199/4*(FNF(I)+2)
180   PSET (X,Y)
190 NEXT
```

List 2.7b

```
100 ' List 2.7b
110 '
120 SCREEN 0,0 : CLS 3
130 DEF FNF(X)=X+SIN(5*X)
140 WINDOW (-2,-2)-(2,2)
150 '
160 FOR I=-2 TO 2 STEP .05
170   PSET (I,FNF(I))
180 NEXT
```

前にも述べましたが，グラフィック関係の命令のうち，PSET，PRESET，LINE，CIRCLE，PAINT，POINT 命令は，座標をワールド座標系で与えます．このうち PAINT 命令以外は実際に表示されない座標を指定しても問題ありません．PAINT 命令に限っては，実際に表示されないような座標を指定するとエラーになります．また，座標の指定をスクリーン座標系で行う命令としては，GET 命令と PUT 命令があります．なお CLS　2 では，VIEW 命令で指定された範囲のみを消去するので，他のプログラムを実行するときは，必ず，SCREEN 0 で初期化してください．

キャラクタ座標

ここまではすべてグラフィック座標についての説明を行ってきましたが，PC-8801 mk IIで使用する座標にはもう 1 種類，キャラクタ座標があります．これも簡単に説明しておきましょう．

文字や数字を表示する位置は，キャラクタ座標によって指定されます．キャラクタ座標の範囲は WIDTH 命令で決めた x 軸方向の文字数で変化してしまいます．例えば WIDTH　80，25 を実行した後には，キャラクタ座標の範囲は x 座標が 0〜79，y 座標が 0〜24 になるわけです（本書では，通常このモードを用いています）．

このとき，グラフィック座標での座標の範囲は，x 座標が 0〜639，y 座標が 0〜199 ですから，8 ドット（グラフィック座標）で 1 キャラクタ（キャラクタ座標）という関係になっています（*List 2.8*，*Figure 20* 参照）．

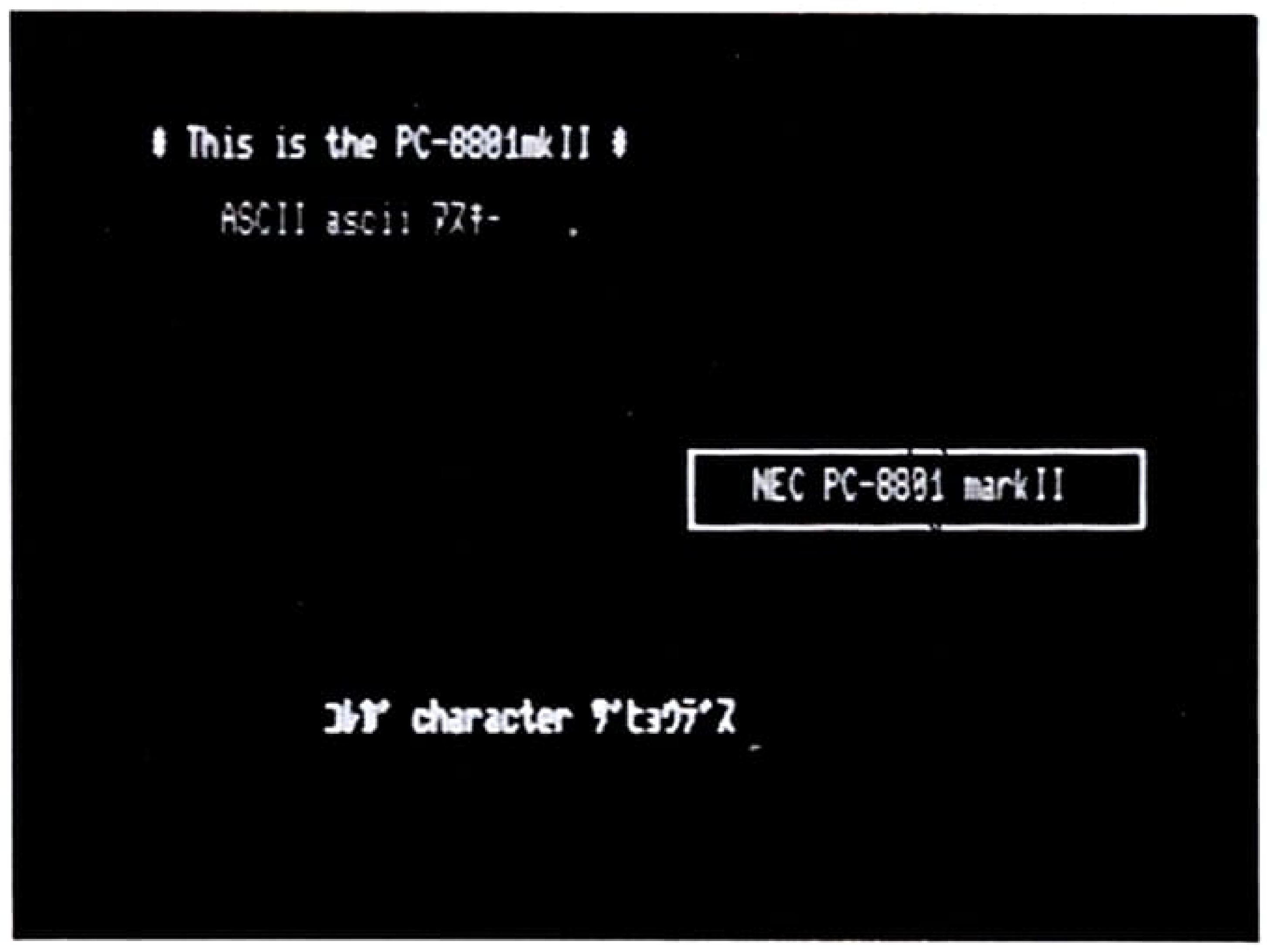


Figure 20

List 2.8

```
100 ' List 2.8
110 '
120 WIDTH 80,25 : CONSOLE 0,25,0,1 : CLS 3
130 COLOR 5: LOCATE 10,2 : PRINT "* This is the PC-8801mkII *"
140 COLOR 6: LOCATE 10,4 : PRINT "     ASCII ascii ｱｽｷｰ "
150 COLOR 7: LOCATE 40,10: PRINT "┌──────────────────────────┐"
160          LOCATE 40,11: PRINT "│                          │"
170          LOCATE 40,12: PRINT "└──────────────────────────┘"
180 COLOR 3: LOCATE 44,11:PRINT "NEC PC-8801 markII"
190 COLOR 7: LOCATE 20,17: PRINT "ｺﾚｶﾞ character ｻﾞﾋｮｳﾃﾞｽ"
```

絵をつくる

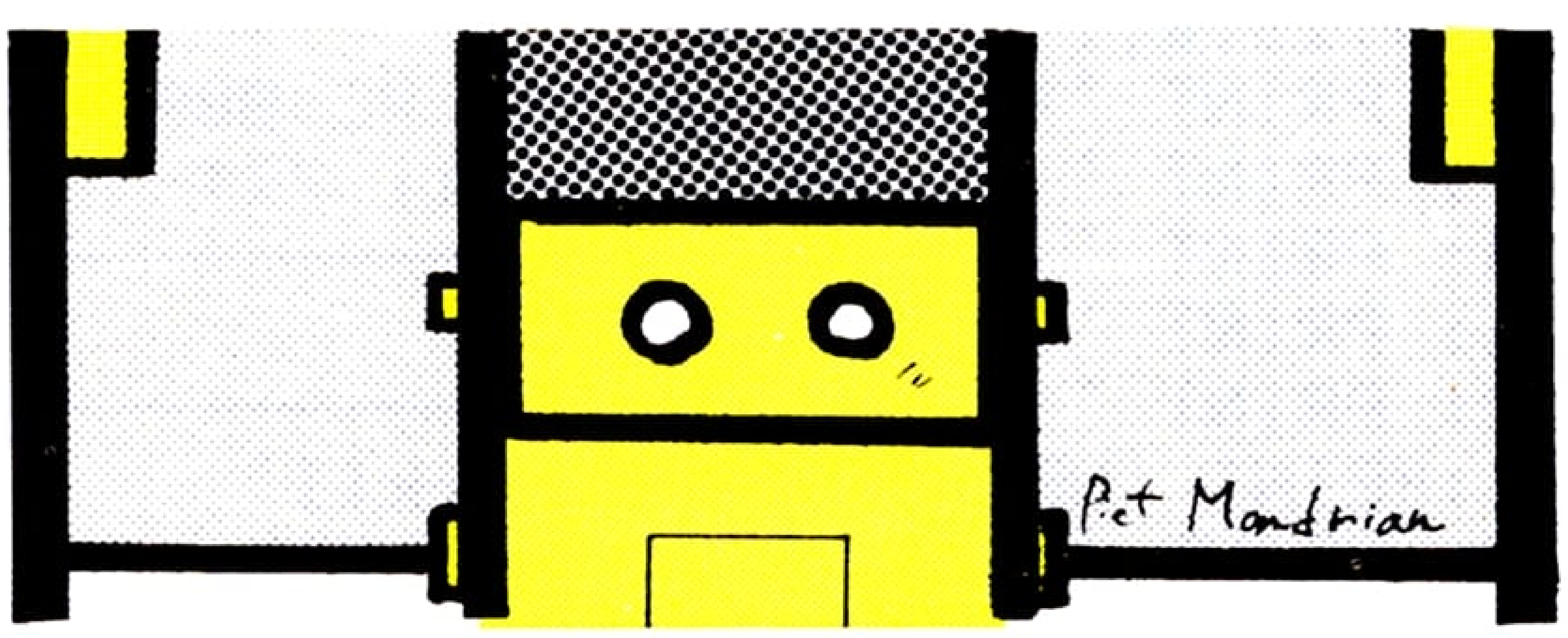
……というわけでありまして，前節では主に「座標の取扱い」ということで話を進めました．座標といえば，これはもう中学高校で苦しめられた（あるいは今現在，苦しめられている真っ最中の）数学のことを思い浮かべる方も多いことでしょう．そうです，前節で私たちは数学していたのです．その冷たくて味もそっけもない非人間的かつ学生すべての心労のもと？ である数学の世界から離れ，ここでグラフィックの芸術的可能性について眺めてみましょう．

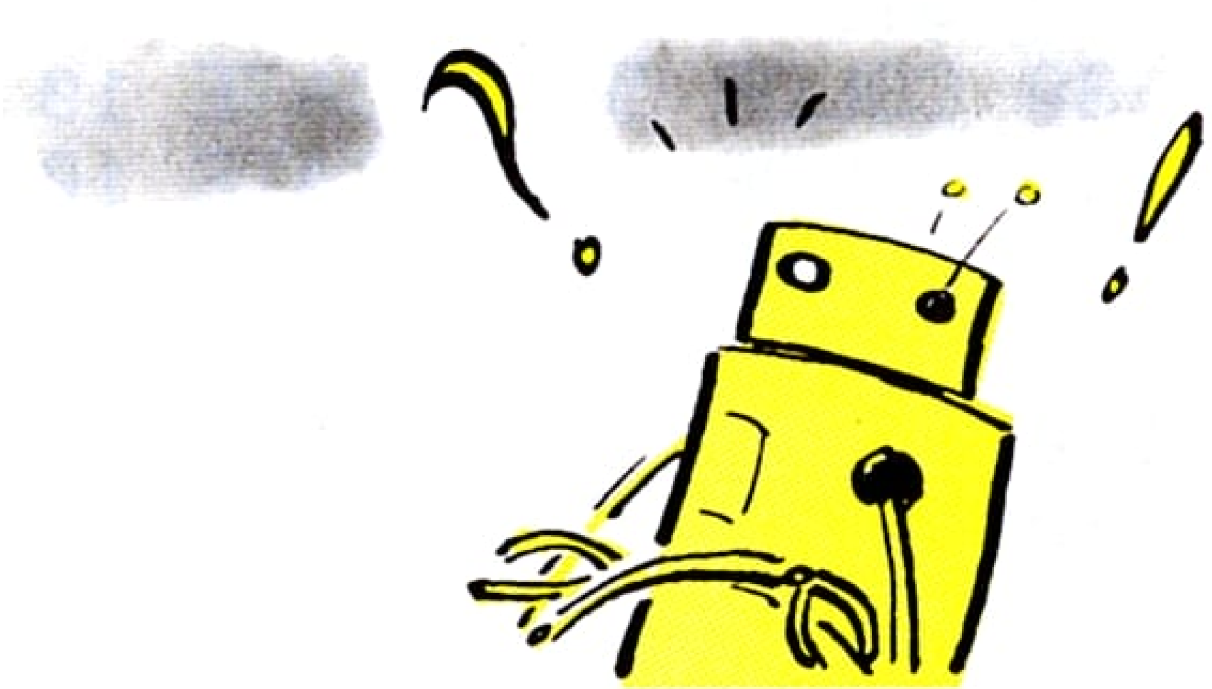
本節で問題になるのは数字を扱う能力よりも，むしろいかにして美しい絵を描き出すか（ほら，同じグラフィックの表示結果でも前節では「図形」といっていましたが，今や「絵」といい換えられました），要するにあなたの絵心，美的センス，といったものが大切になってきます．

あっ，不安そうな顔をしている人がいますね．なに，図画工作は昔から２しかもらったことがないって？ いえ，そんな心配はご無用．コンピュータ・グラフィックスの良いところは，手先の器用さが作品のできばえに関係しないという点です．器用な人が鮮やかにタイプしたものであろうが，ブキッチョな人が１文字１文字打ち込んだものであろうが，プログラムさえ同じであればPC-8801 mk IIは常に同じ動作をしてくれます．

今まで絵なんて描いたことないよ，という人にとっても，コンピュータ・グラフィックスは恰好の道具として，そして大いなるインスピレーションの源となって，きっとあなたの眠れる感性を引き出してくれるでしょう．

と，話はとてつもなく大きくなってしまいましたが，要は，せっかくのコンピュータ，きれいなグラフィックを楽しみましょうということ．そのために知っておいてほしいことのいくつかを，本節で紹介いたします．





コマンドを並べて

絵を描く? 何も考えることないでしょ.線を引くならLINE命令,円を描くのはCIRCLE命令,必要なだけいくらでも並べればいいわけでしょ——といういい分は確かに正しい.それが証拠にこんな絵を作ることもできます.*Figure 21* を見てください.これは *List 2.9* のプログラムを実行した結果です.

このようなプログラムを作るには,まず方眼紙に下絵を描いて,必要な座標をひろっていくとよいでしょう.普通のグラフ用紙を使うと縦横比を考えなければならないのでやっかいです.ここはひとつ,始めからPC-8801 mk IIの座標に合わせた縦長マス目のデザインシートを使いたいですね.Appendixにこの PC-8801 mk II 専用方眼紙を載せましたので,ご入用の際には,コピーしてお使いください.

座標をひろう際に,注意しなければならないのは円の半径の決め方です.円の半径はx軸方向のドット数で指定します.縦横の比率を変えて描く場合にも基本となるのはx軸方向のドット数ですから間違えないように.

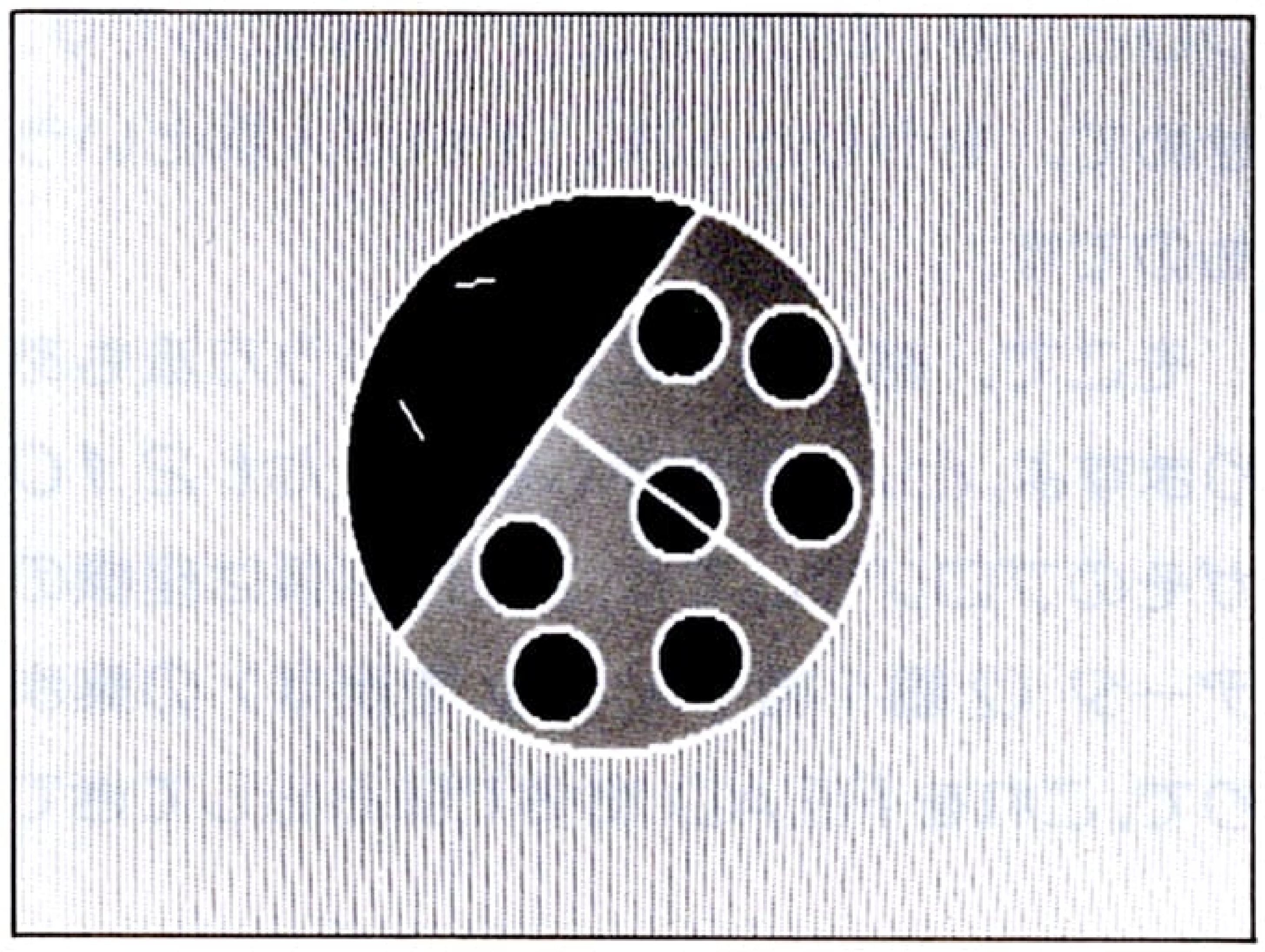


Figure 21

List 2.9

```
100 ' List 2.9
110 '
120 SCREEN 0,1 : CLS 3
130 CIRCLE (300,100),120,7
140 CIRCLE (330,70),20,7
150 CIRCLE (380,75),20,7
160 CIRCLE (390,105),20,7
170 CIRCLE (260,120),20,7
180 CIRCLE (275,144),20,7
190 CIRCLE (340,140),20,7
200 CIRCLE (330,108),20,7
210 LINE (230,60)-(247,59),6
220 LINE (205,85)-(215,93),6
230 LINE (205,135)-(340,43),7
240 LINE (276,89)-(400,132),7
250 PAINT (300,110),2,7
260 PAINT (110,20),CHR$(&H22)+CHR$(&H44)+CHR$(&H66),7
```

READ DATAの利用

コマンドを並べて絵を描くという前述の方法，何かもの足りないと思った人はいませんか．その人はなかなかスルドイ，将来有望です．いったいコマンド-ズラズラ方式のどこがもの足りないのか．それは単純すぎるというところです．BASICの命令としてあらかじめ用意されている命令のひとつひとつを，そのまま使っているからです．せめて，もうひと工夫，それが人間というものです．

そこで，絵を描くにはこういう方法もありますよ，という例を示します．*List 2.10* がそのプログラムです．200行以後は座標のデータ（2個1組でxとyを表す）の集まりで，これをREAD命令で読み出してきて直線を引きます（*Figure 22* 参照）．

160～180行に注目．FOR～NEXTループで，「あること」を57回繰り返しています．その「あること」というのは，READ命令でXとYの座標を読み出してきて現在の位置からその（X，Y）に向けて直線を引

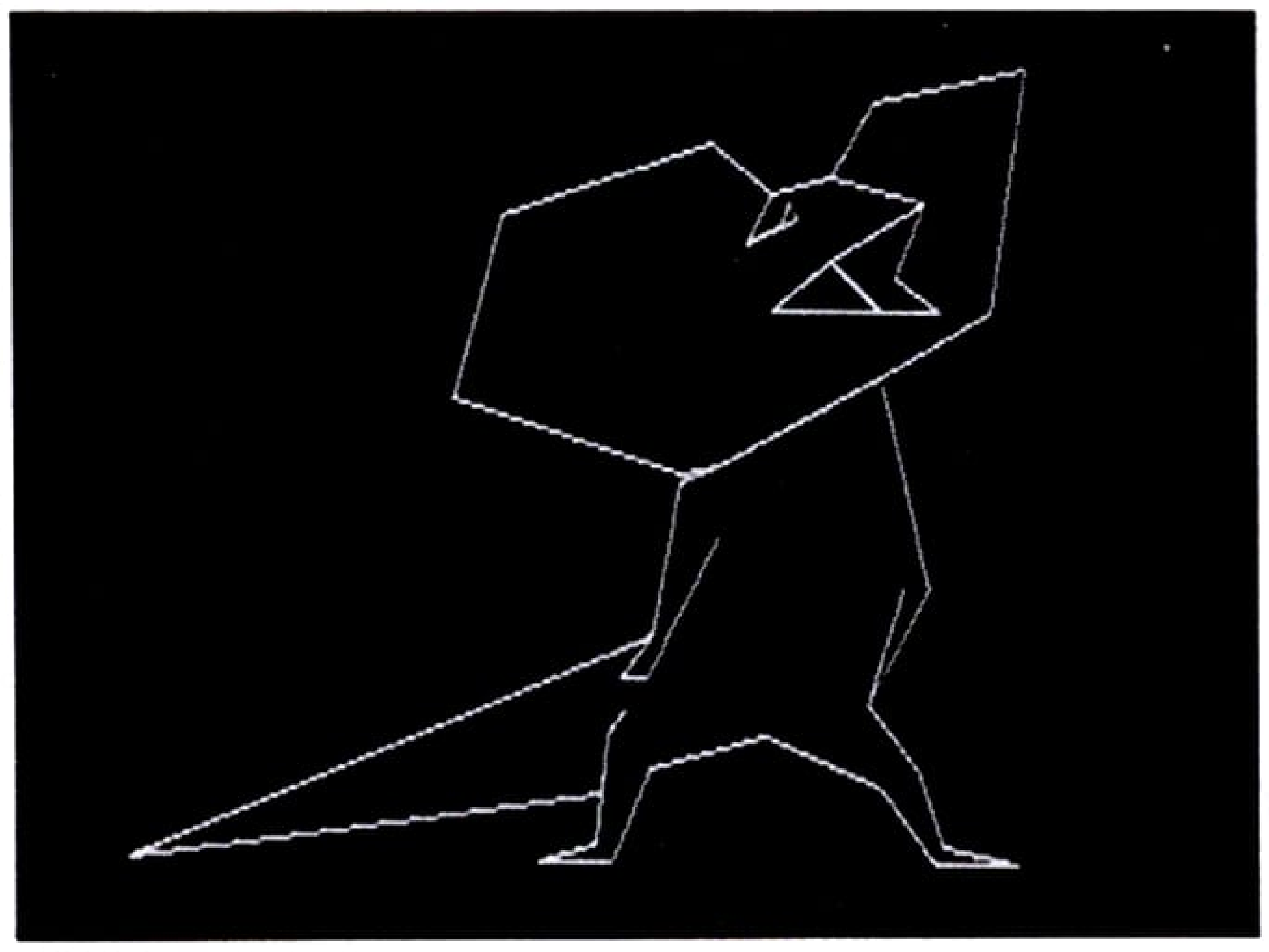


Figure 22

List 2.10

```
100 ' List 2.10
110 '
120 SCREEN 0,0 : CLS 3
130 C=5
140 RESTORE 200
150 READ X,Y : POINT (X,Y)
160 FOR I=1 TO 57
170   READ X,Y : LINE -(X,Y),C
180 NEXT I
190 '
200 DATA 144,24, 150,18, 164,16, 174, 7, 210, 3, 202,32, 176,40
210 DATA 188,66, 174,80, 182,66, 174,80, 186,88, 192,97, 210,99
220 DATA 190,99, 177,90, 150,84, 123,88, 115,99,  98,99, 111,97
230 DATA 113,84, 117,81, 113,84, 112,91,   4,99, 123,72, 116,77
240 DATA 122,77, 138,60, 122,77, 116,77, 123,72, 129,53, 138,51
250 DATA 176,40, 138,51, 129,52,  76,43,  87,21, 136,12, 150,18
260 DATA 144,24, 152,22, 154,19, 156,21, 152,22, 144,24, 150,18
270 DATA 164,16, 186,19, 179,28, 190,32, 150,32, 164,26, 176,32
280 DATA 164,26, 186,19
```

く，という作業です．LINE 命令で，「始点の座標を省略すると前回引いた直線の終点が今回の始点として使われる」という機能を，非常にうまく利用しています．こうしていわゆる一筆書きが描けるわけです．

同じ一筆書きでも，LINE 命令を並べるのに比べてこちらはプログラムに柔軟性があります．例えば，175 行に C=7−C を加えてみましょう．直線が 1 回引かれるごとに，色が赤，水色と交互に変わっていきます．こんなことはコマンドを並べていてはできない芸当です．

話が前後しますが，ここで 150 行を見てください．これは次の 160 行から始まる一筆書きに備えて，最初の点を決めています．この部分がないと，電源を入れた直後の「暗黙の始点」は（0，0）の位置になっているため画面左上から線が引かれてしまいます．試してみてください．

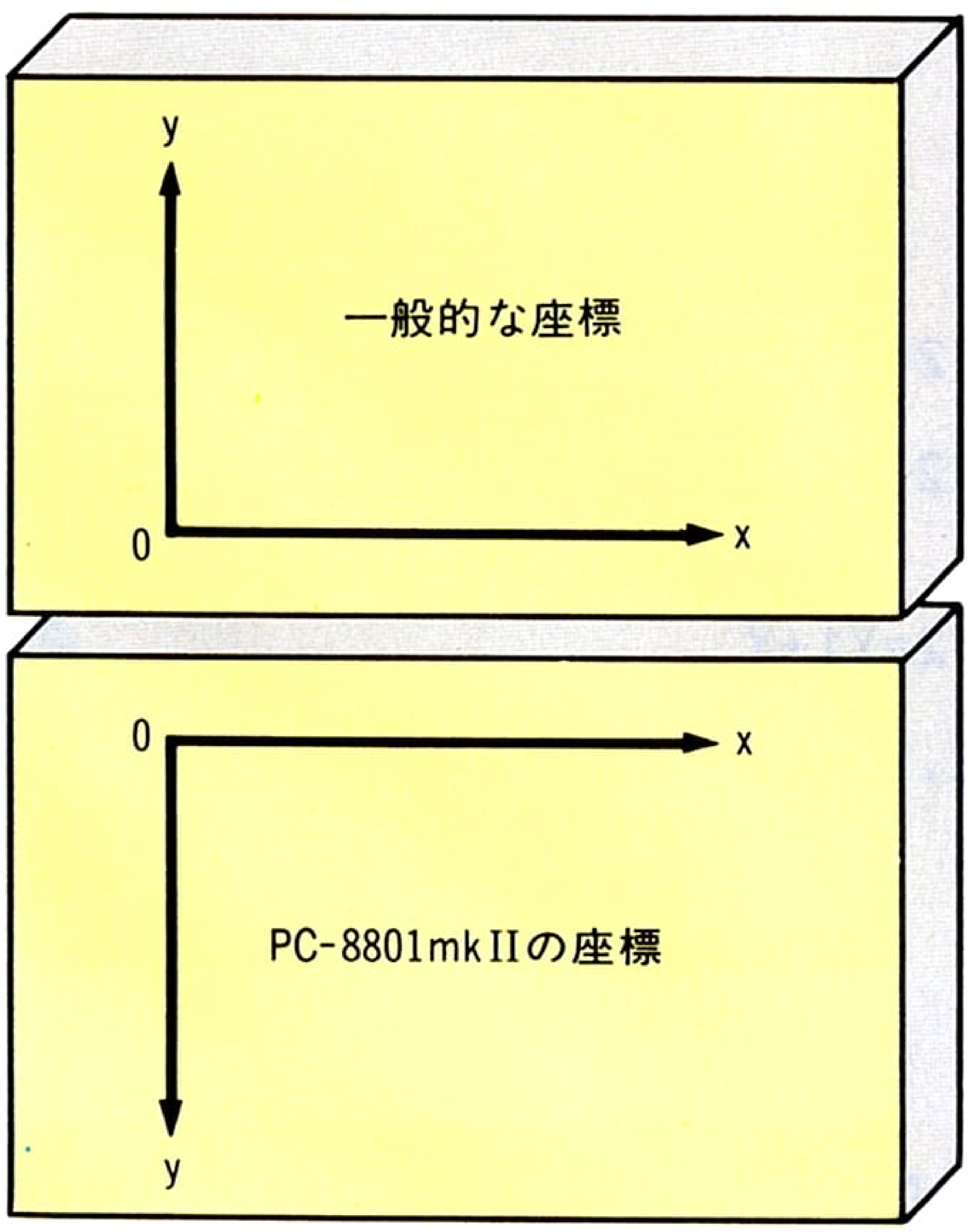


Figure 23

角度と距離による位置の指定

自分で何から何までデザインしたグラフィックを画面に表示させてみるのも楽しみ方のひとつですが，コンピュータが独自に計算して作り出したグラフィックをボンヤリと眺めている，これもひまつぶしにはもってこいの遊びです．

もとは単純な図形でも（そう，たった 1 本の直線でも），コンピュータの計算能力によって変化し，組み合わされていく様子は，その整然としたリズムの中に思いがけない美しさが発見できて楽しいものです．

このような計算によるグラフィックは次章でもいろいろと扱っていますが，1 つだけここで説明を加えておきたいことがあります． sin 関数と cos 関数，例の三角関数というヤツの使い方です．

あれ，もう数学しないんじゃなかったの．やだよ，sin，cos なんて……．

まぁまぁ，そんなこと言わないで．難しいことはやりませんから．要するに高さを求めるのが sin 関数，横幅を求めるのが cos 関数ってこと．知ってりゃいいのはこの 2 つだけなんです．円を基本にしたグラフィックには，sin 関数と cos 関数が必要不可欠であるといいきって間違いなし．多少頭が痛くても，これを覚えておいて絶対にソンはありません．

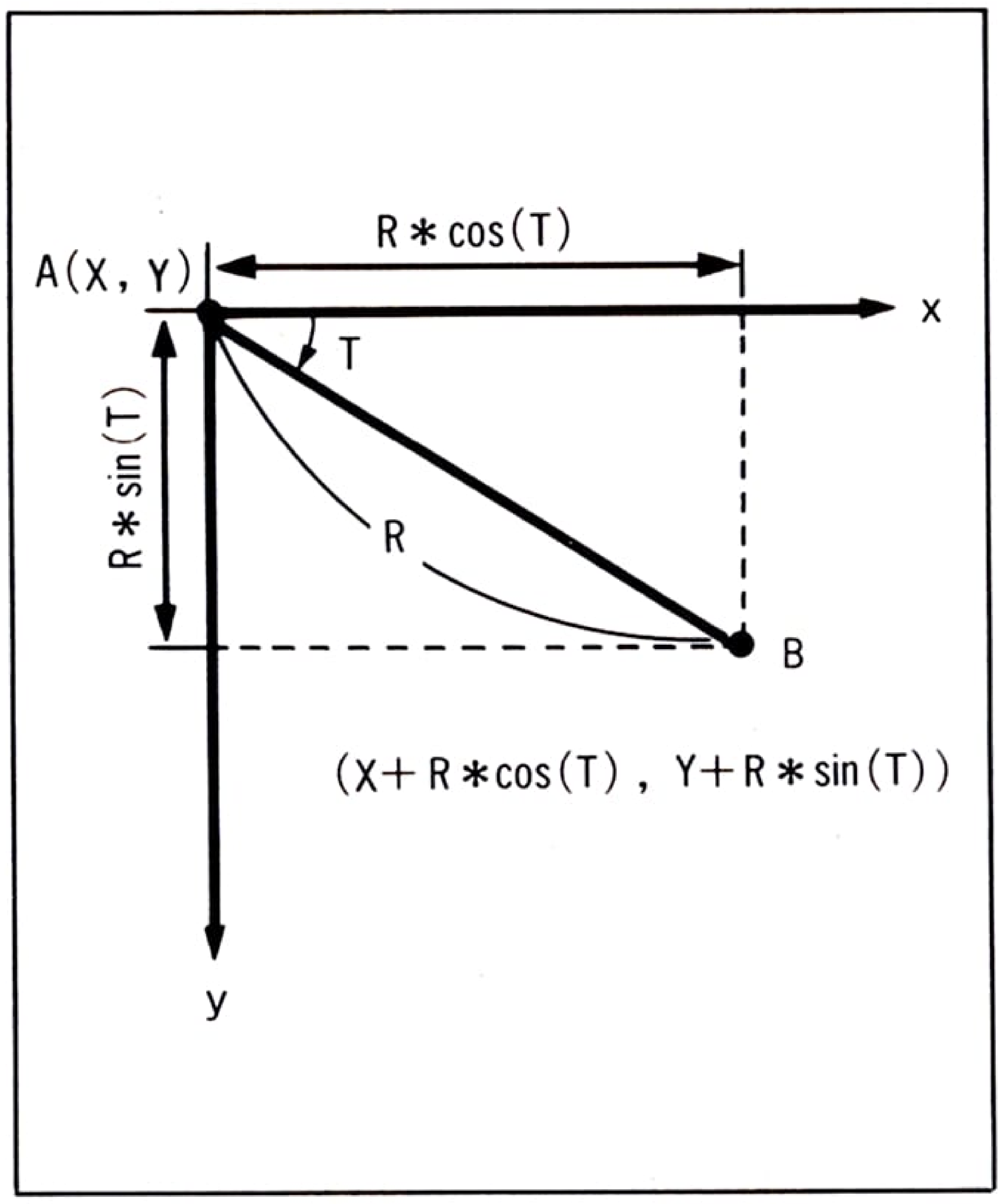


Figure 24

PC-8801 mk IIで使う座標は，x 座標が右方向を，y 座標が下方向を向いています．学校で習ったものと y 軸の向きが逆なので注意してください（*Figure 23* 参照）．

座標（X，Y）にある点 A から直線距離で R，水平方向との角度が T だけ離れた点 B の位置を考えます．B までの横方向の距離，すなわち x 座標の幅は R ＊ cos(T)，そして B までの縦方向の距離，すなわち y 座標の幅は R ＊ sin（T）として計算されます．よろしいですか，横が cos 関数，縦が sin 関数，くれぐれも間違えないように．

例えば R を 10 cm に決めて角度 T を少しずつ変えながら B の位置に点を打っていけば，A を中心とした半径 10 cm の円が描

List 2.11

```
100 ' List 2.11
110 '
120 PI=3.14159 : P=.5 : C=1
130 SCREEN 0,0 : CLS 3
140 FOR T=0 TO 2*PI STEP PI/21
150   X=80*COS(T): Y=80*SIN(T)
160   X1=200 : Y1=100 : X2=X1+X : Y2=Y1+Y
170   LINE (X1,Y1)-(X2,Y2),7
180   CIRCLE (X2,Y2),3,C : PAINT (X2,Y2),C
190 '
200   X=80*COS(T): Y=80*SIN(T)*P
210   X1=400 : Y1=100 : X2=X1+X : Y2=Y1+Y
220   LINE (X1,Y1)-(X2,Y2),7
230   CIRCLE (X2,Y2),3,C: PAINT (X2,Y2),C
240   C=C+2 : IF C=7 THEN C=1
250 NEXT T
```

List 2.11a

```
140 FOR D=0 TO 360 STEP 8.57146 : T=D*PI/180
250 NEXT D
```

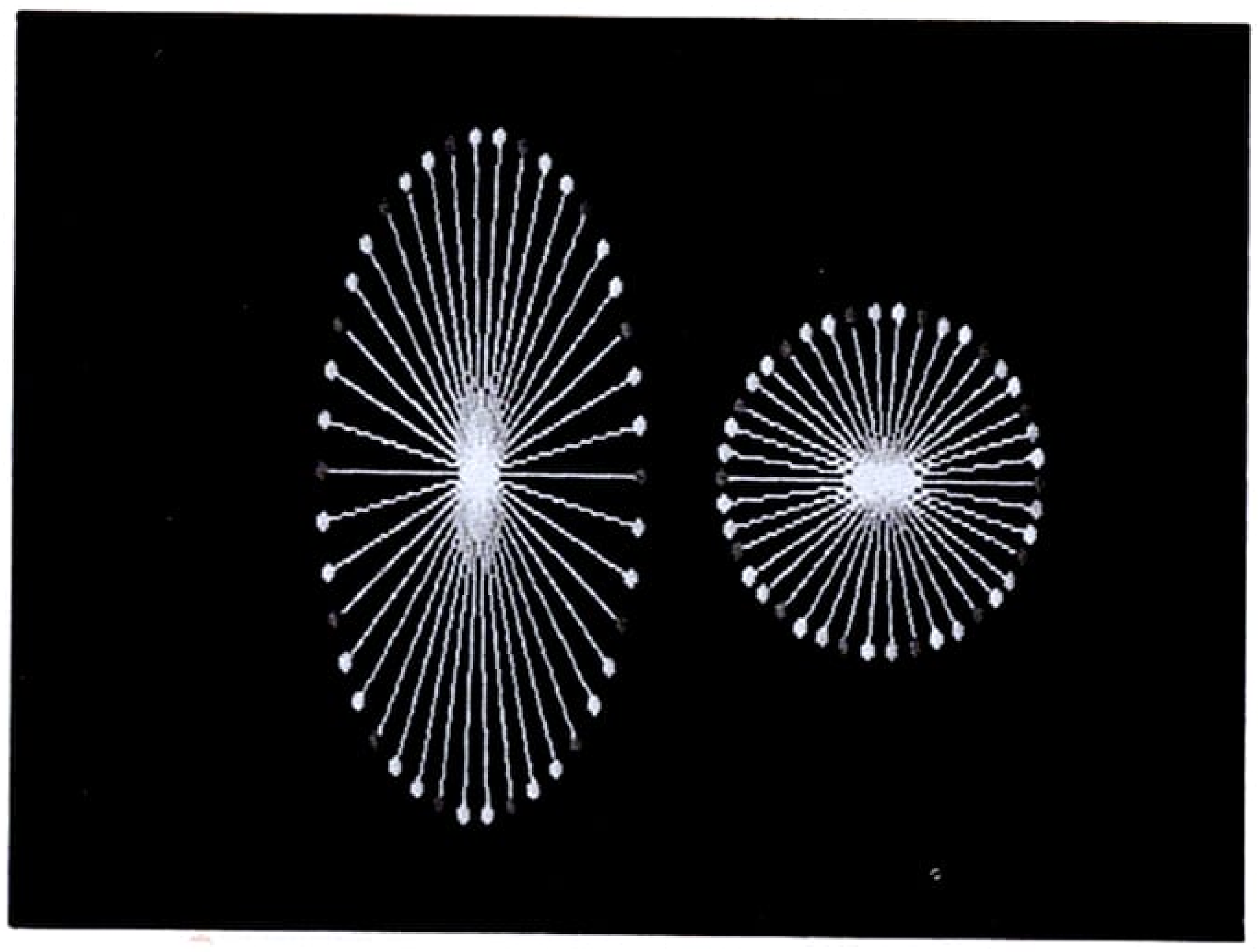


Figure 25

けることになります(*Figure 24* 参照).

ただし前述のようにPC-8801 mk IIではドットの縦横比が違うため，これをそのまま実行すると縦長の円が描かれてしまいます．まともな円にするには，計算された値に0.5を掛けてy座標を修正しなければなりません．*List 2.11* を見てください．画面の左側に表示される円は150行で計算されたy座標をそのまま使っているので縦に長い円となりますが，画面の右側の円は200行でy座標にP(=0.5)を掛けているので真円になります（*Figure 25* 参照).

三角関数で，もう1ついっておかなければならないのは角度の単位のことです．N88-BASICの三角関数は角度をラジアンという単位で指定します．この単位は円周360度を2π(=6.28)ラジアンとして換算するもので，1度がπ/180ラジアンに相当します．D度ならD＊π/180ラジアンです．これは数学的にはとても便利な単位ですが実用的には使いづらいものです．

このプログラムの中では角度Tを0ラジアンから2πラジアンまで，直接にラジアン単位で指定していますが，もしラジアンがわからなければ，140，250行を *List 2.11a* のように変えると，おなじみの“度”を単位としてループを回すことができます．

乱数の利用

乱数は，コンピュータ・グラフィックスに欠かせないスパイスともいうべきものです．乱数を使用する目的としては大まかに，次の2つが挙げられます．

まず1つめは，文字どおりランダムな効果を出すため，つまり，ここかと思えばまたあちら，グラフィックの現れる位置を予期できないおもしろさをねらうというものです．

乱数を使うもう1つの目的は，グラフィックに自然の趣きを与えるということ．森の木々も海辺の砂粒も，遠くから眺めれば皆一様に見えても，実際にはそれぞれが個性を持った形の集まりです．そのような状況を画面上に表現する場合には，コンピュータの規則正しい計算力はかえって不都合，乱数によって得られる散らばりが自然の味を創り出すのです．

プログラム中で乱数を使用するためには，乱数関数RNDを用いるということはもう知っていますよね? このRNDは，0以上1未満の範囲の乱数を得るもので，用途に応じてこれにいろいろと手を加えて，望みの範囲の乱数を得るわけです．

*List 2.12*のプログラムを見てください．210行と220行では，x座標を0～634の整数乱数，y座標を0～194の整数乱数とい

う形で得ています．このように，0 から N−1 までの整数を値とする乱数が欲しい場合には，次の式を用います．

INT (RND＊N)…………a

0 から N−1 までの N 個(注意して！N−1 個ではありません)の整数のうちの 1 つが，この式の値となります．そして，この式を基本として，次のような応用もできます．

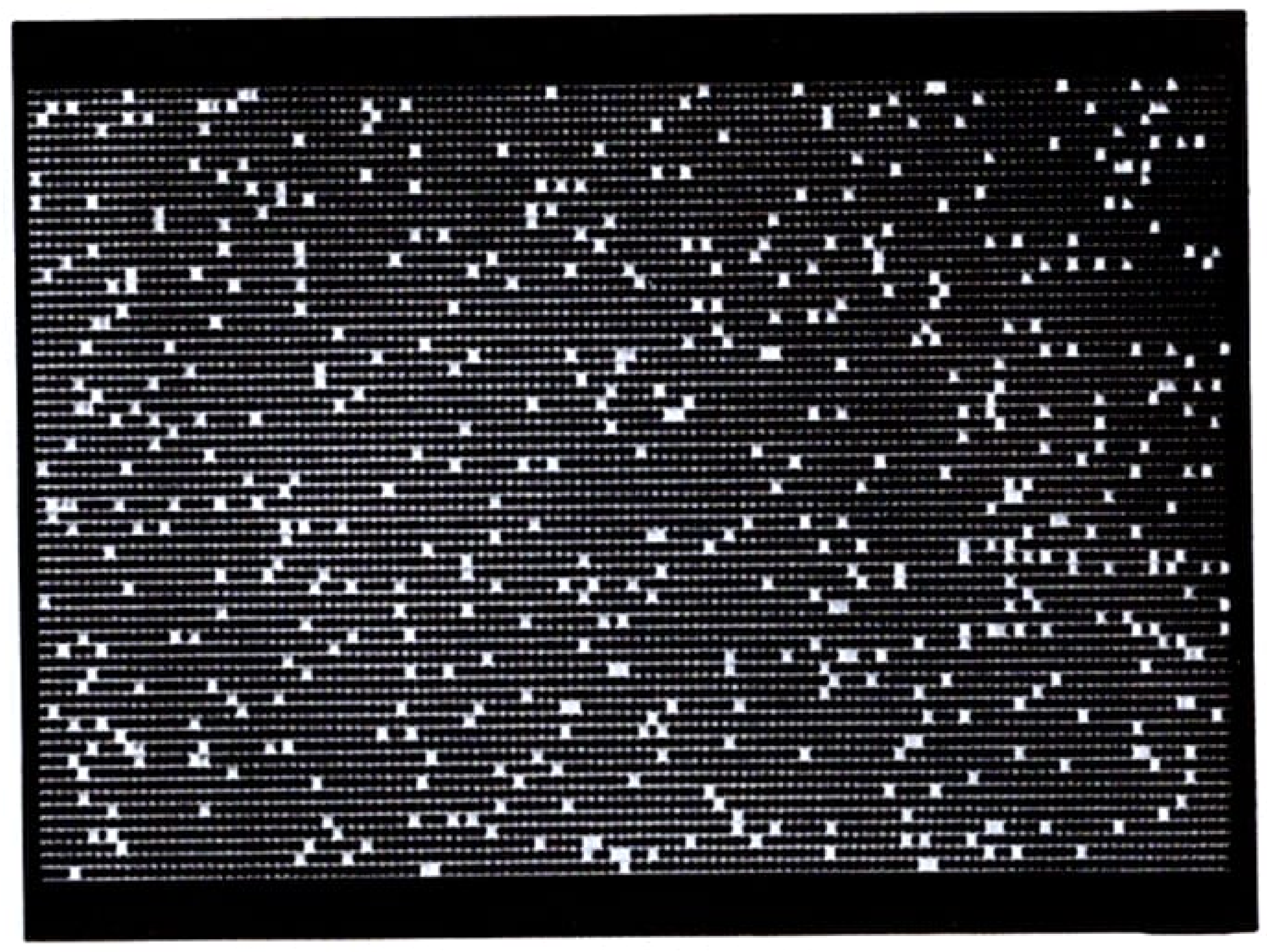


Figure 26

List 2.12

```
100 ' List 2.12
110 '
120 SCREEN 0,0 : CLS 3
130 FOR I=0 TO 635 STEP 5
140   LINE (I,0)-(I,195),1
150 NEXT I
160 FOR I=0 TO 195 STEP 3
170   LINE (0,I)-(635,I),1
180 NEXT I
190 '
200 FOR L=1 TO 1000
210   X=INT(RND*635)
220   Y=INT(RND*195)
230   PAINT (X,Y),RND*2+5,1
240 NEXT L
```

List 2.12a

```
210   X=INT((RND+RND+RND)/3*635)
220   Y=INT((RND+RND+RND)/3*195)
```

List 2.12b

```
210   X=INT(RND*635)
220   Y=INT((RND*RND*RND)*195)
```

● 1から6までの整数をランダムに出す

```
INT (RND*6)+1
```

● 40, 50, 60の3つ数のうち1個をランダムに出す

```
INT (RND*3)*10+40
```

● −2から+2までの整数をランダムに出す

```
INT (RND*5)-2
```

などなど．以下次章でもこの乱数の式は数値の範囲をいろいろ変えて登場します．

ところでa式の乱数は，どの値の現れる確率もN回に1回で等しくなっています．これを，ある特定の数値がよく現れるように手を加える方法もあります．その簡単な例を2つ紹介しましょう．

1つめは，ある範囲の中心部がよく出るようにする方法．これはRNDをN個足します．すると0以上N未満の範囲の乱数が得られますが，出現確率はN／2が最も高くて，両端の0やNに近い値ほど低くなります．ちょうど西洋の鐘（ベル）のような形の確率分布になります．Nを大きくするほど，つまりRNDをたくさん足すほど，中心部の確率が高く，両端は低くなります（*Figure 27*参照）．

*List 2.12*の210，220行を*List 2.12a*のように変えてみてください．星が画

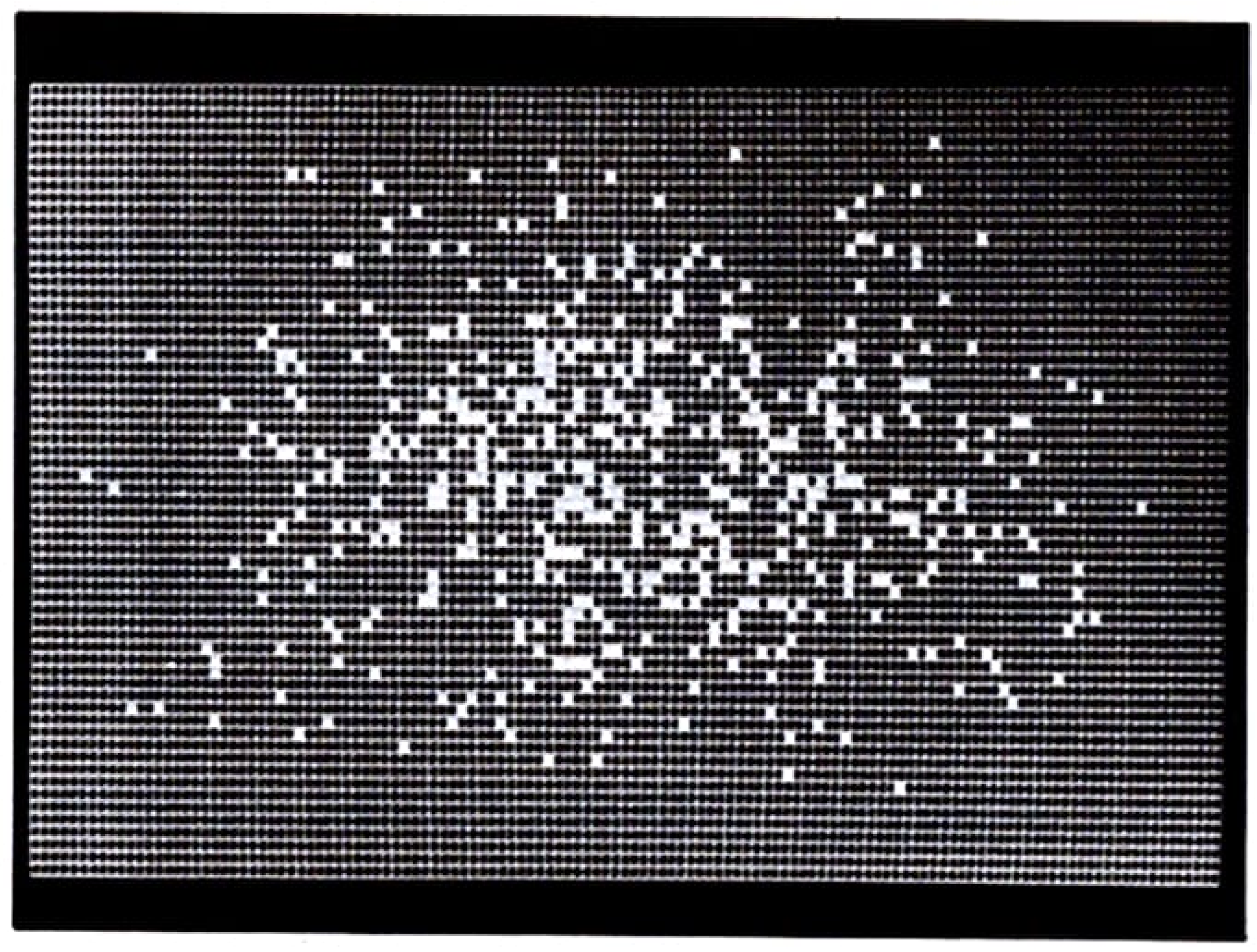


Figure 28

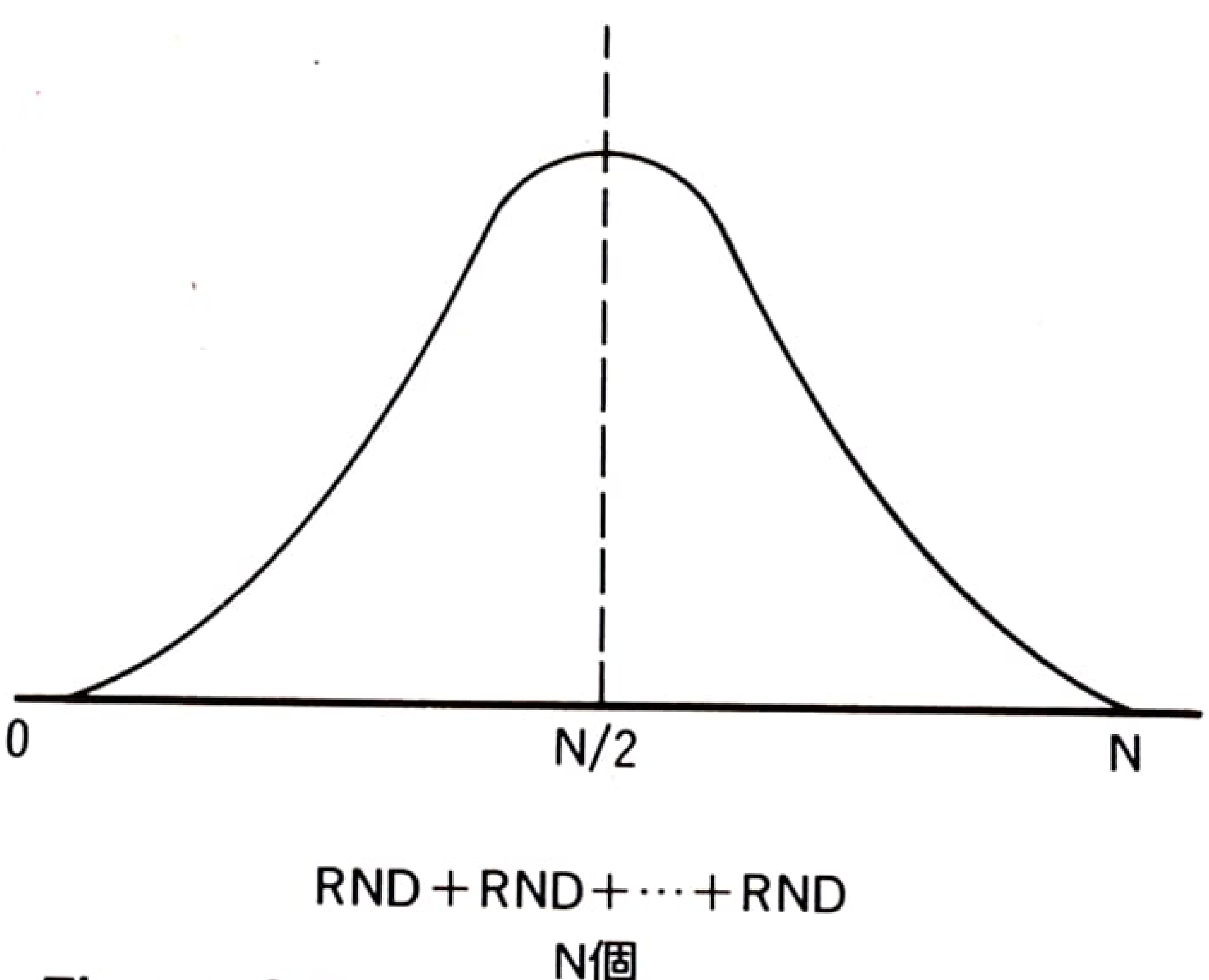


Figure 27

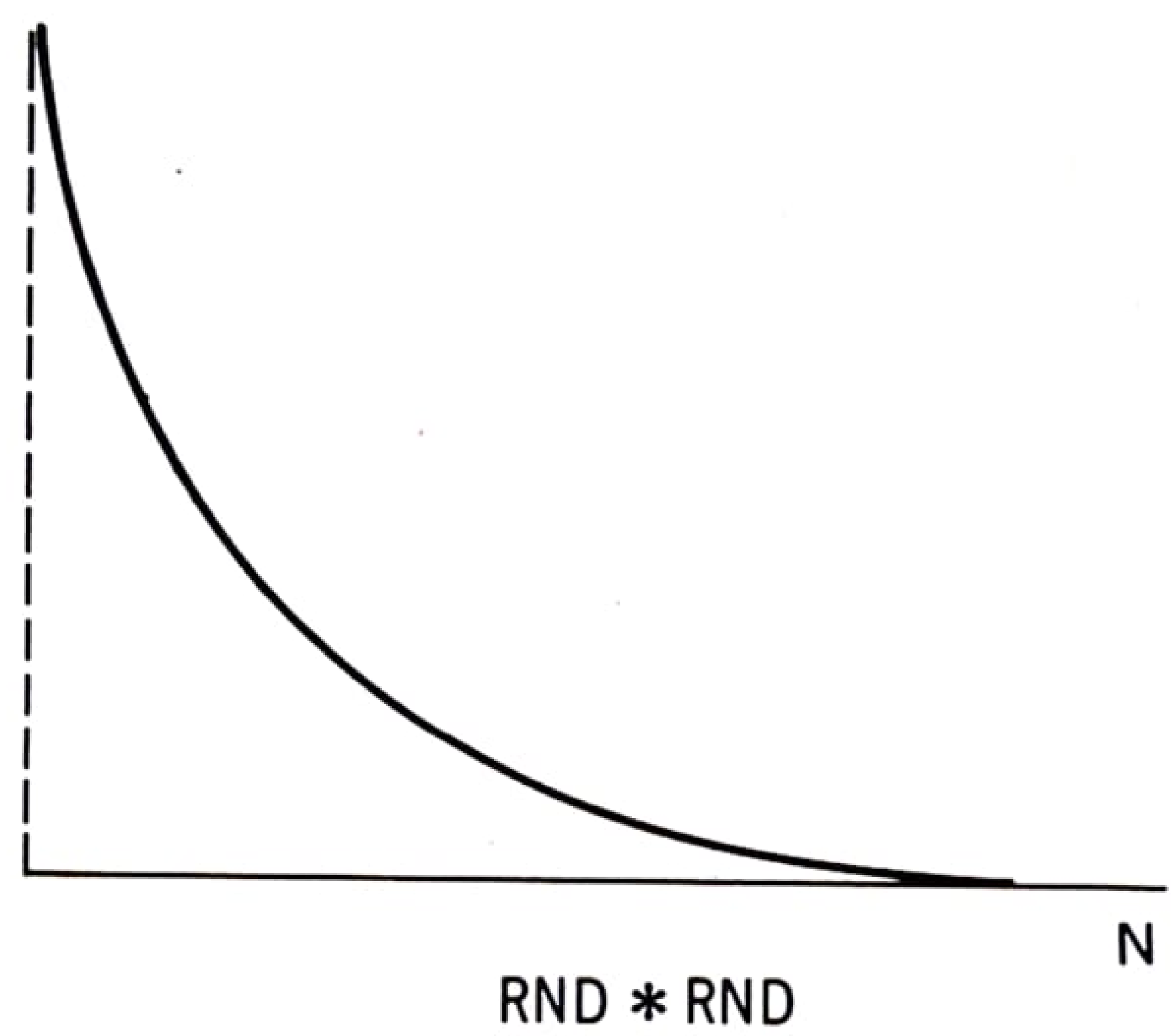


Figure 29

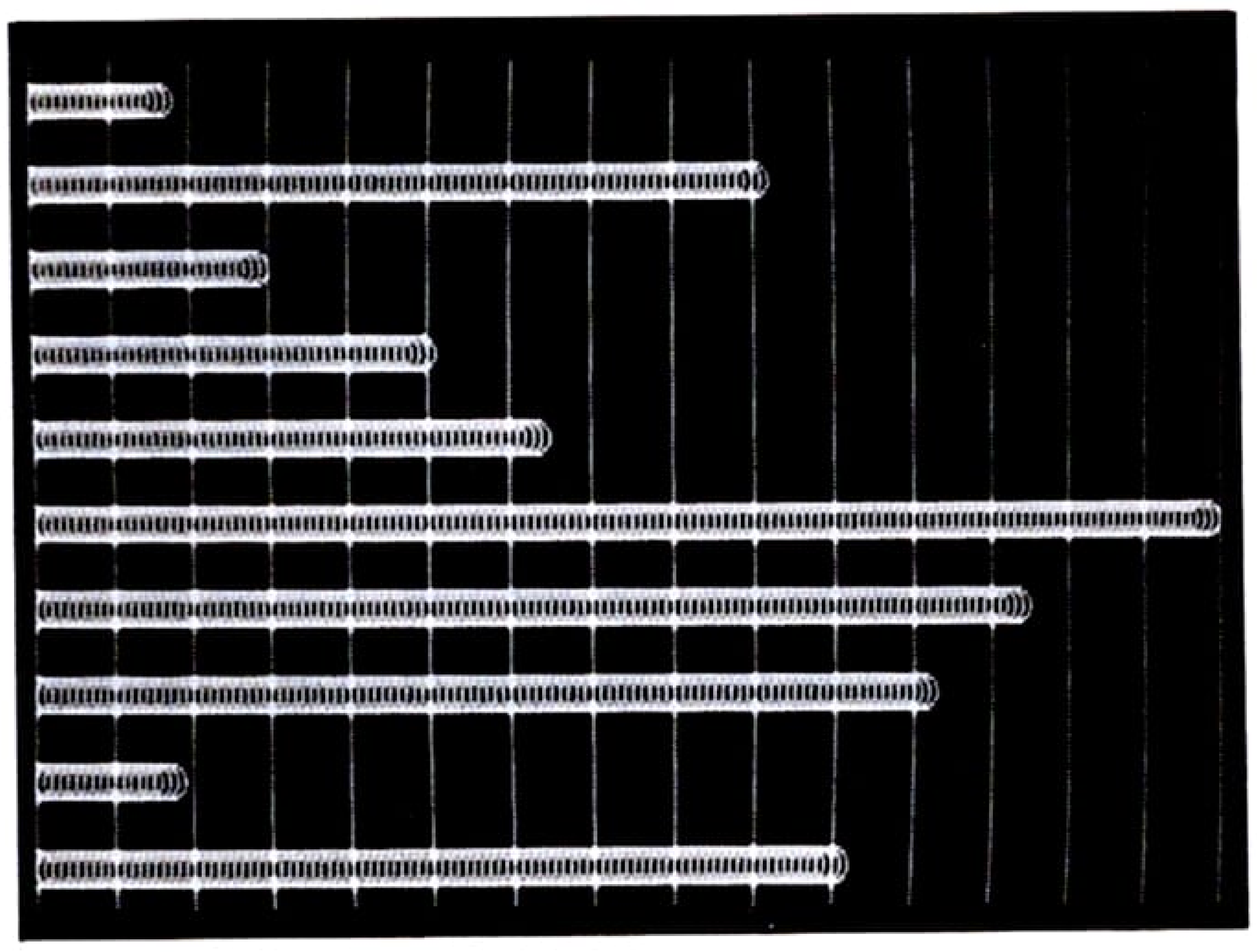


Figure 30

面の中心部に集まってきます．RNDを，3個でなく4個，5個とたくさん足すに従って，星が集まる度合も大きくなります（*Figure 28*参照）．

乱数の確率分布を変えるもう1つの方法は，RNDを掛け合わせるというものです．この場合は，乱数の範囲は0以上1未満で変化しませんが，0に近い部分の出現確率が高く，1の近くはほとんど出ません．つまり，これは*Figure 29*のような形になります．

210行，220行を*List 2.12b*のように変えると星は画面の上側に集まってくるでしょう．

ところで乱数は，何の用意もなしに使うと，プログラムを走らせるたびにまったく同じ順序で値が現れます．例えば，

```
PRINT RND;RND;RND
```

というプログラムを実行してみてください．必ず，.245121 .305003 .311866の順（N_{88}-BASIC Version 1.3）で3個の数値が表示されるはずです．これでは，ゲームなどに乱数を使う場合，毎回同じ局面が現れてきて興ざめということになります．

これを防ぐためにはRANDOMIZE命令を用います．*List 2.13*を実行してください．実行するたびに異なるグラフが現れます（*Figure 30*参照）．もっとも，この例ではRANDOMIZE命令に与える引数にTIME$の秒のみを使っているので，異なるといっても60種類しかありませんが．

List 2.13

```
100 ' List 2.13
110 '
120 SCREEN 0,0 : CLS 3
130 RANDOMIZE VAL(RIGHT$(TIME$,2))
140 '
150 FOR X=0 TO 640 STEP 40
160   LINE (X,0)-(X,199),3
170 NEXT X
180 '
190 FOR Y=10 TO 190 STEP 20
200   X1=INT(RND*620)
210   FOR X=0 TO X1 STEP 4
220     CIRCLE (X,Y),8,5
230   NEXT X
240 NEXT Y
```

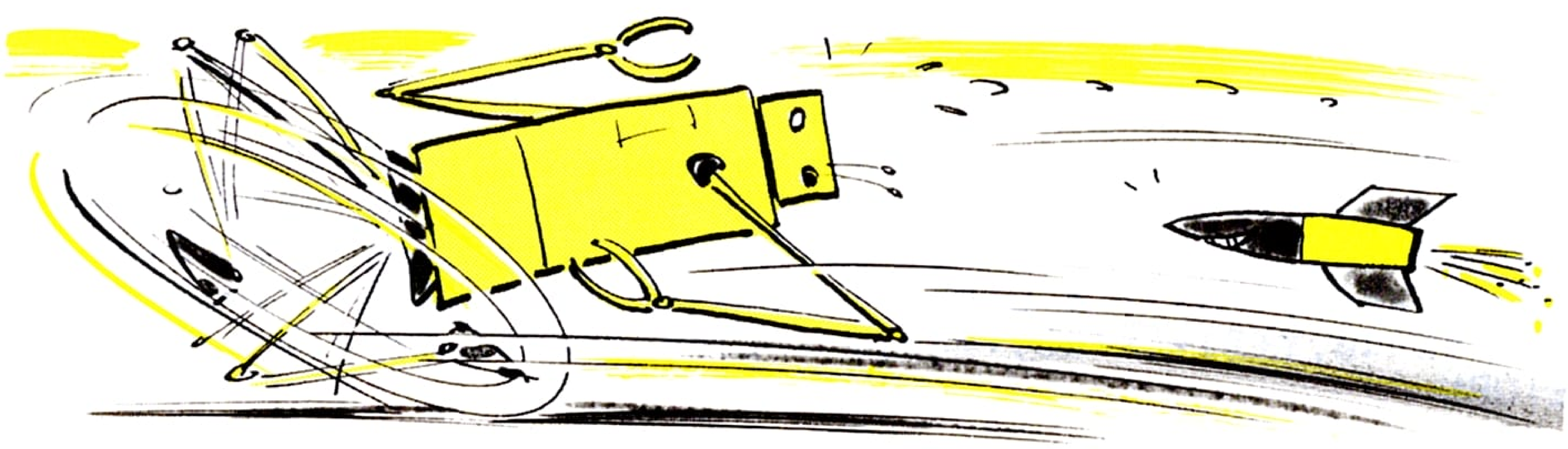
動くグラフィックのこころみ

2月30日午後10時，指定の場所に来られたし…….差出人不明の手紙を受け取った私は，好奇心も手伝って，よせばいいのにノコノコと出かけて行った．ドアを開けるとむやみに長いだけの廊下が奥へと通じていた．コツ，コツ，私の靴音のみが響く．廊下のなかほどの壁に一枚の絵が掛かっていた．

おやおや，ジョコンダのおばさんか．モナ・リザといえば通りはよいが．何たる俗悪趣味だ．そう思いながら歩みを進めたとき，私の視線の端で何か動いたような気がした．はてな，今通り過ぎたあの絵の方から何か気配を感じたが？ 振り向くと絵の中の彼女が私にニヤリと笑いかけた…….

と，やや不気味な雰囲気で本節を始めてみましたが，これは別にこれから説明することと大して関係はありません．ただ，私たち人間は動くものに対して非常に敏感だ，ということをいってみたかっただけなのです．考えごとをしていて目の前の電柱に気づかないことはあっても，その視界の中に犬なり猫なりがとびこんできたら，驚かずにはいられません．いや，これは人間だけではありません．カエルは動くものしかエサだと思わないといわれてますし，卵からかえったばかりのヒヨコは，はじめて見た動く物を母鶏だと思い込むそうです．動きのあるものに興味を覚えるのは，生まれたときから備わった動物の本能というものなのでしょう．

まぁ，とにもかくにも，画面に現れた図形なりキャラクタなりが動き出せば，それは，静止したままの画面よりは格段におもしろくなることは，当然ともいえましょう．それが自分の作ったプログラムであればなおさらでしょう．そこで本節では，グラフィックに動きを与えるというテーマにそって説明していきます．



同じ形を並べて

ここに1匹のネズミがいます．チューと鳴いて1匹増えました．もう1度チューと鳴いてさらに1匹増えました．横1列に並んで，はいおしまい．

なんだ，これは数が増えているだけで，ネズミはちっとも動いていないじゃないかって？ いやまだ始まったばかりなんですよ．そんなに早く目的を達成してしまってはつまらないでしょう．こういう地道な努力の積み重ねが大切なんですよ，ホントに（*Figure 31* 参照）．

さて，このプログラムはどのように作られているのでしょうか．*List 2.14* を見てください．190行からが，1匹分のネズミを描くサブルーチンです．

この中の座標はすべて，（X，Y）の位置を基準として，右に何ドット左に何ドットという形で表され，基準となる（X，Y）の値を変化させれば，画面のどの位置にでもネズミが表示できるようになっています．位置をある1か所に固定してしまうのではなく，基準となる座標からの相対的な変位として与えるこのような方式を，相対座標による位置指定といいます．

ネズミが横1列に並ぶのは，130行でｙ座標を固定し，140〜160行のループでｘ座標

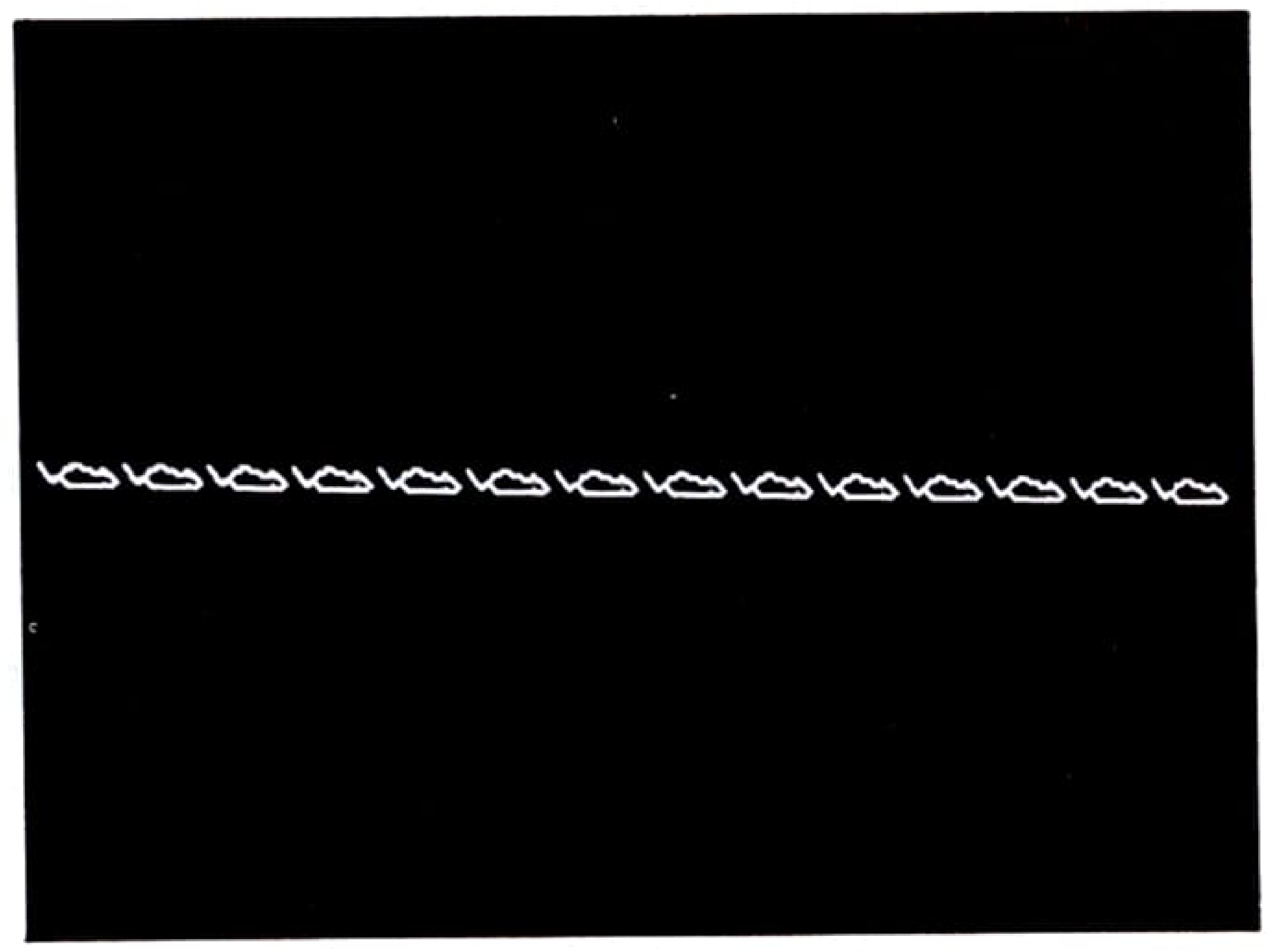


Figure 31

List 2.14

```
100 ' List 2.14
110 '
120 SCREEN 0,0 : CLS 3
130 Y=105
140 FOR X=0 TO 585 STEP 45
150   GOSUB *MOUSE : BEEP 1 : BEEP 0
160 NEXT X
170 END
180 '
190 *MOUSE
200 LINE (X+ 7,Y+0)-(X+13,Y+4),7 : LINE -(X+18,Y+3).7
210 LINE -(X+20,Y+1),7 : LINE -(X+26,Y+0),7 : LINE -(X+38,Y+2),7
220 LINE -(X+40,Y+1),7 : LINE -(X+43,Y+2),7 : LINE -(X+48,Y+4),7
230 LINE -(X+45,Y+5),7 : LINE -(X+22,Y+5),7 : LINE -(X+18,Y+3),7
240 LINE (X+40,Y+3)-(X+40,Y+4),2 : LINE -(X+41,Y+4),2
250 RETURN
```

を次々と変化させているからです．逆に，x座標を一定にしてy座標を変化させれば，ネズミは縦に積み重ねられます．そしてxとyの両座標を同時に変化させていけば，その変化のしかたによって，円形や斜めに並ばせることもできます．

表示位置が斜めの直線にそって移動するグラフィックの例を*List 2.15*に示します．このプログラムでは，変数Iの変化に伴い，x座標はXSずつ減っていき，y座標はYSずつ増えていきます．その両者を組み合わせた座標の位置が，ちょうど坂をころがる円の中心となっていくのです(*Figure 32*参照)．

なお，160行のLの値を変えると，画面に表示される円の個数が変わるように，プログラムを作ってみました．これはほんのおまけです．

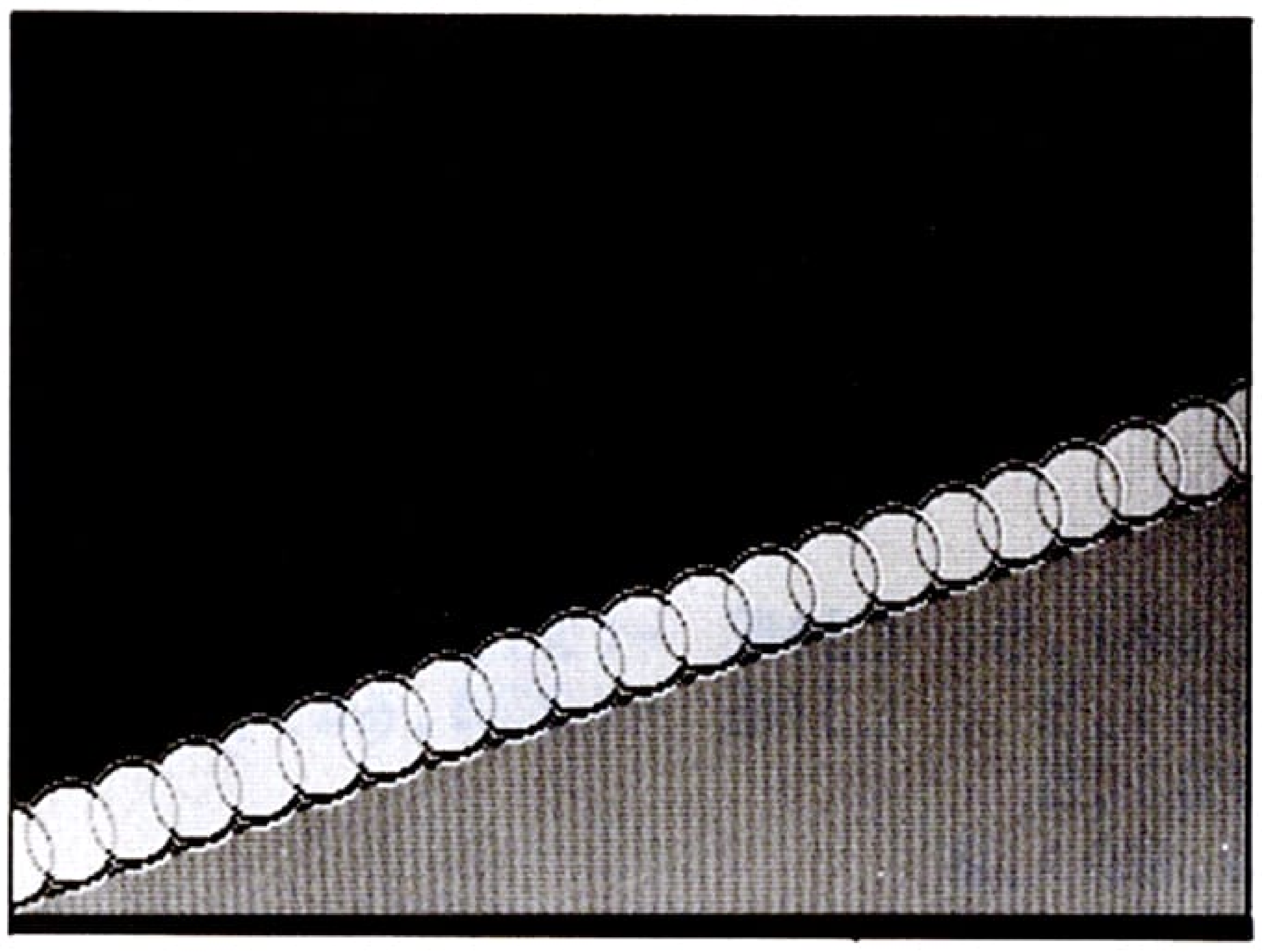


Figure 32

周期的繰り返し

位置の移動だけがグラフィックに与える「動き」ではありません．さてこんなコミカルな信号機はいかが(*Figure 33*参照)．

*List 2.16*のプログラムを実行すると，2つ目玉の黄信号が点滅を始めます．この動作を行わせているのが，230行から290行にかけてのループ部分，その中で重要な働きをしているのがFという変数です．

List 2.15

```
100 ' List 2.15
110 '
120 SCREEN 0,0 : CLS 3
130 LINE (0,199)-(639,199),1 : LINE -(639,100),1
140 LINE -(0,199),1 : PAINT (638,198),1
150 '
160 L=20
170 XS=639/L : YS=100/L
180 FOR I=0 TO L
190   X=639-XS*I : Y=86+YS*I
210   CIRCLE (X,Y),25,6
220   CIRCLE (X,Y),22,2 : PAINT (X,Y),3,2
230   BEEP 1 : BEEP 0
240 NEXT I
```

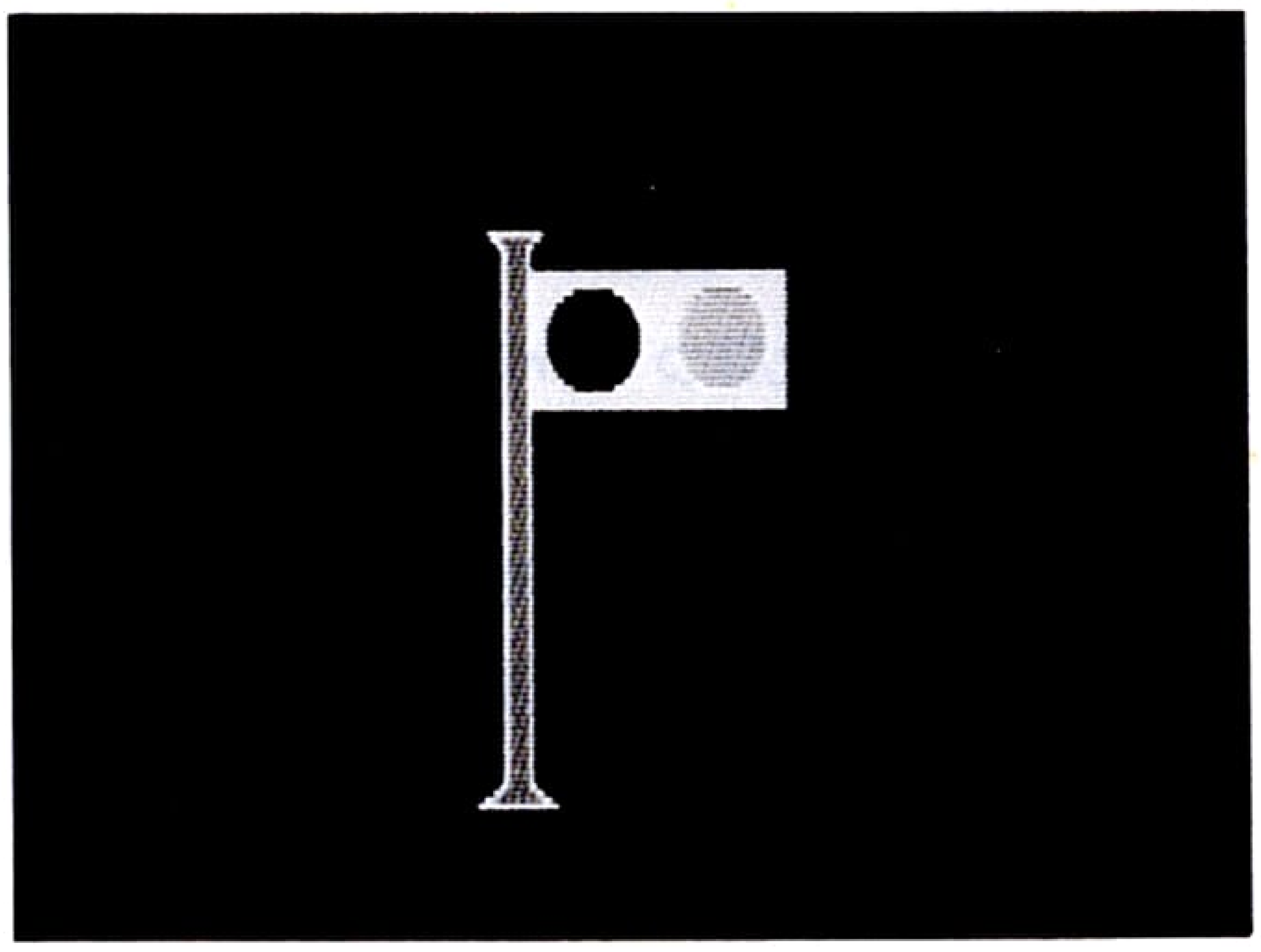


Figure 33

Fには，最初220行で1という値を与えておきます．次の数行分の説明はちょっと後回しにして，280行を見てください．Fの値はここで変化します．今，1であったFの値は，この行の命令を実行すると0になります．ループによって，再びこの280行の命令が実行されると，Fはまた1に戻ります．結局ループを回るたびに，Fの値は1と0が交互に変化するのです．

このFの値によって，どちらの目玉が点

List 2.16

```
100 ' List 2.16
110 '
120 SCREEN 0,0 : CLS 3
130 LINE (320,70)-(400,90),5,BF
140 LINE (320,90)-(320,146),5
150 LINE -(328,149),5 : LINE -(304,149),5: LINE -(312,146),5
160 LINE -(312,66),5  : LINE -(308,64),5 : LINE -(324,64),5
170 LINE -(320,66),5  : LINE -(320,70),5
180 CIRCLE (340,80),14,0: PAINT (340,80),0
190 CIRCLE (380,80),14,0: PAINT (380,80),0
200 PAINT (318,70),1,5
210 '
220 F=1
230 BEEP 1 : BEEP 0
240   IF F=1 THEN PAINT (340,80),6,5
250   IF F=0 THEN PAINT (380,80),6,5
260   FOR T=1 TO 200 : NEXT T
270   PAINT (340,80),0,5 : PAINT (380,80),0,5
280   F=1-F
290 GOTO 230
```

List 2.16a

```
220 X=340
230 BEEP 1 : BEEP 0
240   PAINT (X,80),6,5
250   FOR T=1 TO 200 : NEXT T
260   PAINT (X,80),0,5
270   X=(380+340)-X
280 GOTO 230
290 REM
```

灯するのか決めているのが240行と250行．しばらく時間をおいた後，270行で黒く塗りつぶします．ここで，点灯している方の目玉だけ消せばよいのですが，両方の位置を黒く塗りつぶしています．これはちょっと手抜き．

このプログラムでは，変数に1と0とを繰り返させましたが，他のどんな数を繰り返させることもできます．変数XにMとNの2数を繰り返させる式は，次のとおりです．

X=(M+N)−X

始めXにMという値を代入しておけば，この式を1回実行するごとに，Xの値はMからNへ，NからMへと交互に変化します．例えば，このプログラムの220行以降の行を *List 2.16a* のように変えてみましょう．これでも結果は同じです．終了するときは STOP キーを押してください．

この場合は，点灯する目玉のx座標，340と380を直接に繰り返させているのです．

移動するグラフィック

では今度は，グラフィックが本当に位置を移していくプログラムを作ってみましょう．これは何のことはない簡単な原理でして，基本的には *List 2.14* のネズミと同様に次々と位置を変えながら絵を描いていけばよいのです．ただし，新しい位置に絵を出す前に，古い位置の絵を消すこと．そうすれば，その絵はどんどん先へ進んでいくように見えます．

List 2.17 の，240行以降のサブルーチンが，走っている人間の絵を描きます．ここでも相対座標を使ってどこに描くかを指定できるようにしています．また変数Cの値によって，ランナーの色を変えることもできます．

160〜210行のループで，ランナーを移動させています．2人のランナーを表示するのは170行と180行，上のランナーを速く先へ進ませるため，170行のランナーのx座標は180行のランナーよりも大きな値になるようにしています．

この2人を消すために，190行と200行が実行されます．ランナーを描くサブルーチンを調べてみると，ランナーの絵を構成するドットの中で，最も左のx座標はX+6，最も右のx座標はX+34，そしてy座標は上の端がY+2，下の端がY+26，この範囲内をすべて塗りつぶしてしまえば，(X，Y)の位置にいるランナー1人を消すことができるのです．上端がY+2である，というのは，上部に円があるのでやや大ざっぱな計

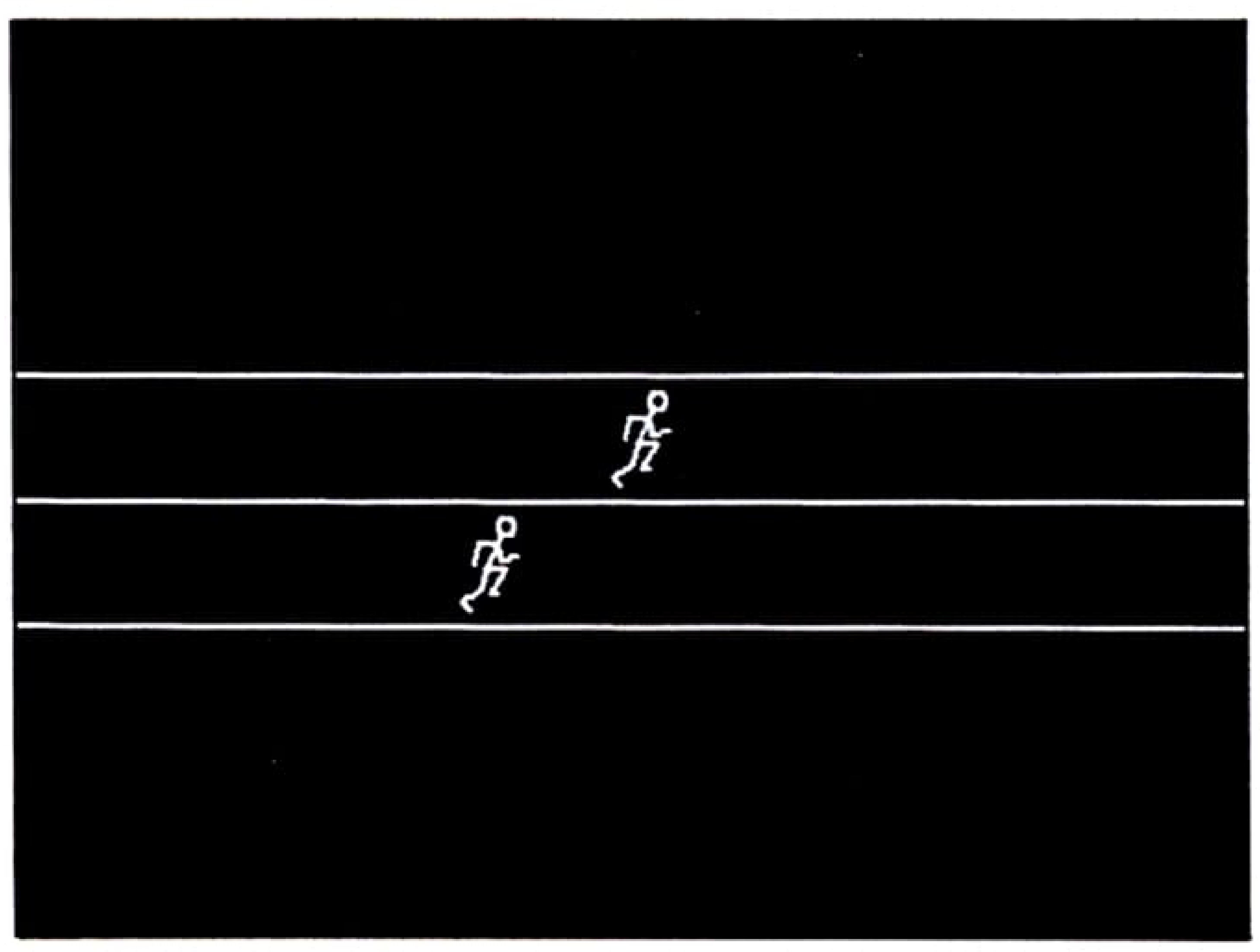


Figure 34

算ですが，ちゃんとランナーが消えてくれれば問題はありません(*Figure 34* 参照)．

この *List 2.17* のプログラムでは，現在表示している位置の絵を消してから，次の新しい位置に絵を出すという順序での移動の様子を示しました．この方法ですと一瞬，画面上からすべての絵が消えている時間があることになります．これではサビシイというので，次に，画面上には常に少なくとも1個の絵が表示されている方法を見てみましょう．

List 2.18 のプログラムでは，220 行からのサブルーチンが，画面上(X，Y)の位置に1台の車を描きます．これをどのように動かしていくのでしょうか？

120 行で決めている（XO，YO）は，現在の車の位置です．140 行で FOR～NEXT

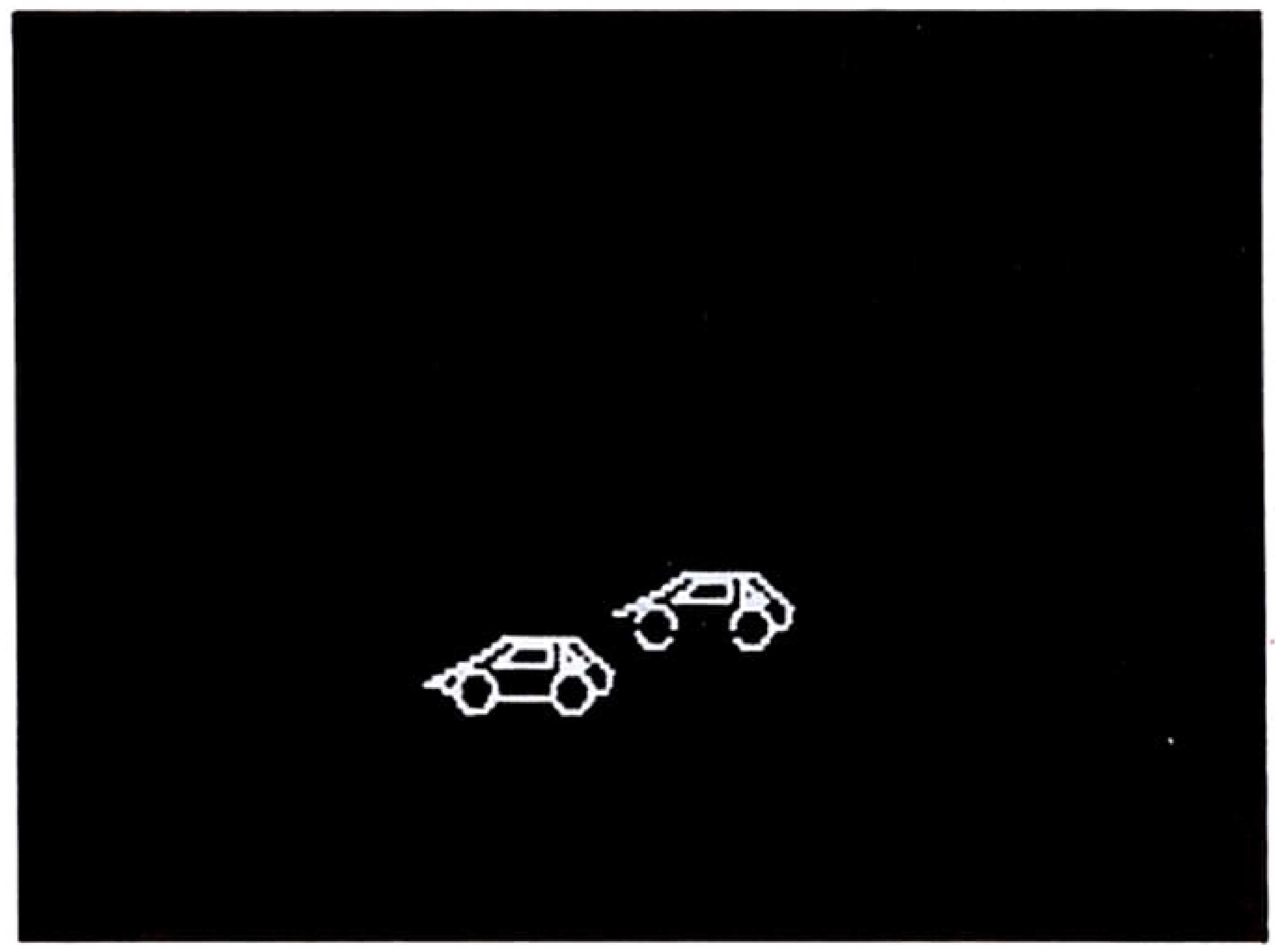


Figure 35

List 2.17

```
100 ' List 2.17
110 '
120 SCREEN 0,0 : CLS 3
130 FOR Y=50 TO 110 STEP 30
140   LINE (0,Y)-(639,Y),7
150 NEXT Y
160 FOR I=0 TO 30
170   X=I*20 : Y=50 : C=3 : GOSUB *MAN
180   X=I*15 : Y=80 : C=5 : GOSUB *MAN
190   X=I*20 : Y=50 : LINE (X+6,Y+2)-(X+34,Y+26),0,BF
200   X=I*15 : Y=80 : LINE (X+6,Y+2)-(X+34,Y+26),0,BF
210 NEXT I
220 END
230 '
240 *MAN
250 CIRCLE (X+28,Y+6),4,C
260 LINE (X+26,Y+8)-(X+18,Y+16),C
270 LINE (X+12,Y+15)-(X+14,Y+10),C : LINE -(X+24,Y+10),C
280 LINE             -(X+25,Y+14),C : LINE -(X+34,Y+13),C
290 LINE (X+18,Y+16)-(X+16,Y+21),C : LINE -(X+6,Y+24),C
300 LINE             -(X+10,Y+26),C
310 LINE (X+18,Y+16)-(X+28,Y+16),C
320 LINE             -(X+20,Y+22),C : LINE -(X+24,Y+22),C
330 RETURN
```

のループに入りますが，150 行で計算している（XN，YN）は，車が次に位置する座標です．さてそこで，160 行でまず次の位置に車を表示してから，170 行で今いる場所の車を消します．そして 180 行では，さきほど計算した新しい位置の座標が，新しい「現在」の座標として受け渡されます．あとは同じことを 150 行から繰り返させることで，車は移動していきます（*Figure 35* 参照）．

たいへんやっかいな作業ですが，こうすれば画面上から絵が消える瞬間はまったくありません．以上の理屈，のみこめましたか？

キー入力による処理

List 2.19 を実行すると，画面の中央に紫色の小さな四角形が現れます．2，4，6，8 のキーを押すと，四角形が動いて跡を残していきます．また 5 のキーを押すと，四角形の色が変わります（*Figure 36* 参照）．

これまで扱ってきたグラフィックは，その形がプログラム中ですべて決定され，人間の介する余地はありませんでした．しかし！ ここでついに私たちは自分の手で，グラフィックを意のままに操ることができるようになったのです!?

とにかく，キー入力を判断してそれに応

List 2.18

```
100 ' List 2.18
110 '
120 XO=580 : YO=80
130 SCREEN 0,0 : CLS 3
140 FOR I=1 TO 10
150   XN=XO-58  : YN=YO+10
160   C=7 : X=XN : Y=YN : GOSUB *CAR
170   C=0 : X=XO : Y=YO : GOSUB *CAR
180   XO=XN : YO=YN :  BEEP 1 : BEEP 0
190 NEXT I
200 END
210 '
220 *CAR
230 LINE (X+41,Y+9)-(X+21,Y+9),C
240 LINE (X+11,Y+8)-(X,Y+7),C
250 LINE -(X+17,Y+3),C : LINE -(X+26,Y),C   : LINE -(X+48,Y),C
260 LINE -(X+60,Y+5),C : LINE -(X+58,Y+8),C : LINE -(X+53,Y+8),C
270 CIRCLE (X+16,Y+8),6,C
280 CIRCLE (X+47,Y+8),6,C
290 LINE (X+30,Y+1)-(X+40,Y+1),C
300 LINE  -(X+40,Y+4),C : LINE -(X+22,Y+4),C:LINE -(X+30,Y+1),C
310 LINE (X+44,Y+1)-(X+44,Y+4),C
320 LINE -(X+52,Y+4),C : LINE -(X+44,Y+1),C
330 RETURN
```

じた処理を行うのは，ゲームでも実用でもプログラム中には必ず含まれる部分ですから，この手法を知っておくに越したことはありません．

このプログラムでは，X と Y で四角形を表示する位置を決めています．四角形の大きさは横 5 ドット，縦 2 ドットにしています．そこで，四角形を横に移動させるには X の値をプラスまたはマイナス 5，縦に動かすには Y の値をプラスまたはマイナス 4，にすればよいのです．210 行は X に 5 を足すサブルーチンです．ただし 5 を足すと画面右端からはみ出てしまう(つまり X が 639 を超える)場合には何もしないで戻ります．同様に，220，230，240 行は画面からはみ出さない場合に，それぞれ X−5，Y+2，Y−

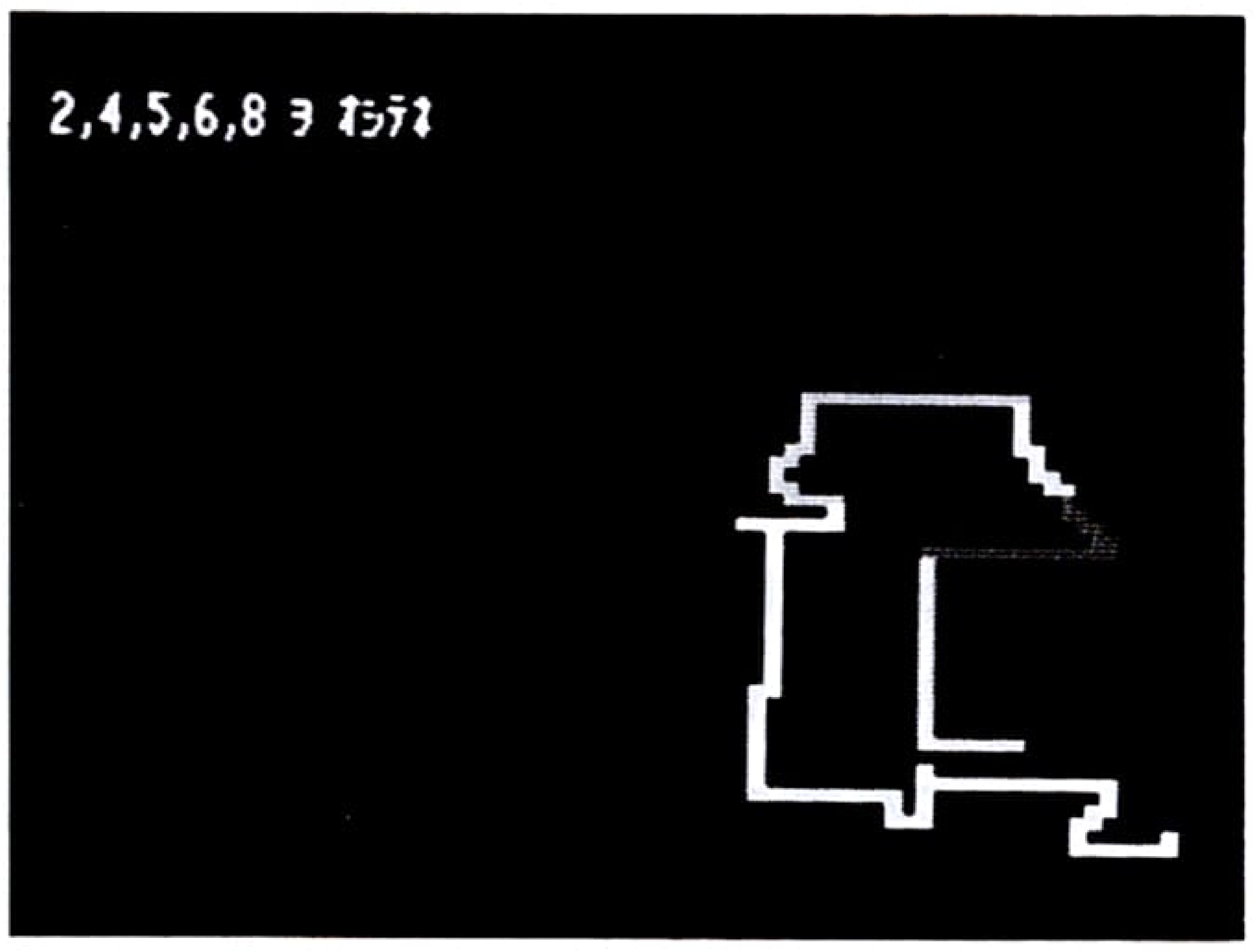


Figure 36

List 2.19

```
100 ' List 2.19
110 '
120 WIDTH 80,25 : SCREEN 0,0 : CLS 3
130 PRINT "2,4,5,6,8 ｦ ｵｼﾃﾈ "
140 X=320 : Y=100 : C=3
150 KY$="64285"
160 '
170 LINE (X,Y)-(X+4,Y+1),C,BF
180   ON INSTR(KY$,INPUT$(1)) GOSUB *RT,*LT,*DN,*UP,*CL
190 GOTO 170
200 '
210 *RT: IF X+5>639 THEN *RE ELSE X=X+5 : GOTO *RE
220 *LT: IF X-5<0   THEN *RE ELSE X=X-5 : GOTO *RE
230 *DN: IF Y+2>199 THEN *RE ELSE Y=Y+2 : GOTO *RE
240 *UP: IF Y-2<0   THEN *RE ELSE Y=Y-2 : GOTO *RE
250 *CL: IF C=7 THEN C=1 ELSE C=C+1
260 *RE: RETURN
```

List 2.19a

```
130 PRINT "ｶｰｿﾙ ｷｰ ｦ ｵｼﾃﾈ"
150 KY$=CHR$(28)+CHR$(29)+CHR$(31)+CHR$(30)+"5"
```

2を行うサブルーチンです．

250行は四角形の移動とは関係ありません．これは，色を決める変数Cを操作しています．それまでC=7だったなら，このサブルーチンの実行で次はC=1となり，そうでなければCは1だけ増えます．結局Cの値はこのサブルーチンを1回実行するごとに，1→2→3→4→5→6→7→1…と変化していきます．

そして，押されたキーによって上の5つのサブルーチンの中から1つを選び出して実行させている，プログラムの心臓とでもいうべき部分が180行です．今，どのキーが押されたのか，それはINPUT$（1）を調べるとわかります．INSTR関数を用いると，その文字が文字列KY$（ここでは64285）の何番目にあるかがわかります．さらに，その何番目の文字であるかという順番によって，ON〜GOSUB命令が行き先を振り分けてくれるのです．たった1行ですが，この180行は非常に中身の濃い仕事をしているんですよね．

180行のINSTR関数の中の文字列KY$を他のものに変えれば，別のキーの組み合わせで移動方向を指定することもできます．例えばカーソルキーで動きを操作するならば，プログラムを *List 2.19a* のように変更してください．終了するときは，STOP キーを押してください．

文字列を150行で定義し直して，180行でこのKY$を "64285" の代わりに使えばよいでしょう（カーソルキーを文字として扱うには，CHR$(28)〜CHR$(31)を使うのですが，ご存知でしたか？）．

位置の判断

グラフィックが今，画面上のどこに位置しているのか，この判断は意外と重要な問題です．ある領域から外へ出してはいけない物体をどうすれば閉じこめておけるのか．アキレスが亀のシッポに触れたかどうか調べるにはどうするか．要するにすべて座標の問題だといってしまえばそれまでですが，グラフィックは上下左右の幅を持っているため，簡単ではありません．間違いなく位置の判断をするためには，緻密な頭脳で注意深くプログラミングする必要があります．

List 2.20 はその位置判断のささやかな例です．*List 2.20* を実行すると，まず乱数でゴールのラインが引かれ，そこに向かってランナーが走っていきます．ランナーの体の一部分でもゴールラインに接すれば，そこでレースは終わり，BEEP音がします（*Figure 37* 参照）．

230行以降は，ランナーの絵を描くサブルーチンです．これは *List 2.17* のものとまったく同一ですから，それを流用しても

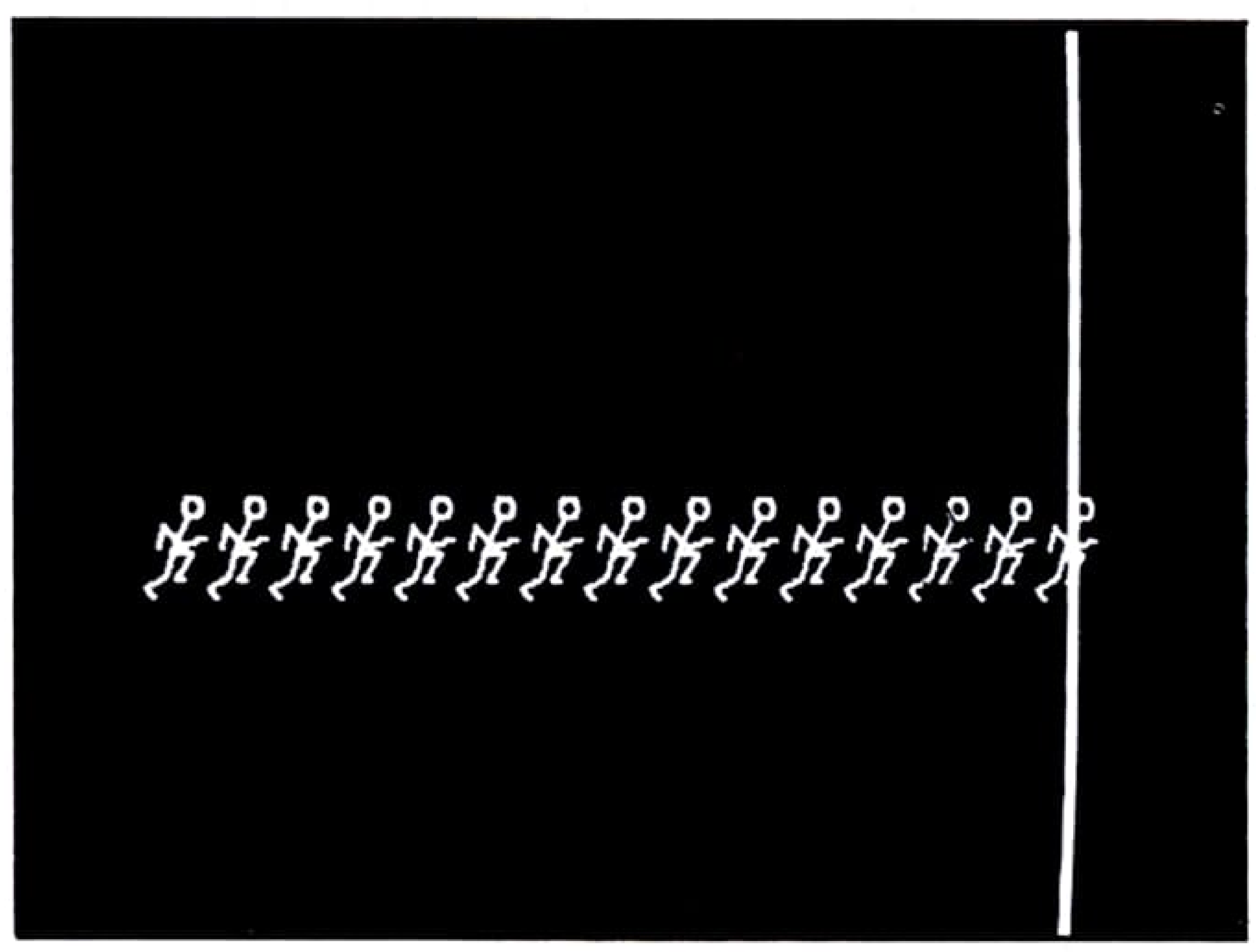


Figure 37

かまいません（前のリストに対して DELETE－230 を実行し，次に RENUM 230 を実行すれば，今度のリストの 230 行以降が一丁あがりというワケ）．ゴールラインを表す x 座標は 140 行で決めて，G という変数に代入しておきます．ランナーが走ってきて，この G の位置を超えたらゴールイン．ランナーの位置は（X，Y）という座標で指定しますから，変数 X の値が G の値を超えたら終わりにすればいいかな，と考えるのはちと甘い．

さきほどもいいましたがランナーのグラフィックは幅を持っています．その左端の x 座標は X+6，右端の x 座標は X+36 です．このグラフィックが一部分でもゴールに接したときというのは，右端の X+36 がゴールに達したときのことをいいます．すなわち，190 行のように X+36 の値が G の値を超えたということをもって，ゴールインを判断しなければならないのです．

それでは応用問題をひとつ．「ランナーの全身がゴールラインを超えた時点で，ゴー

List 2.20

```
100 ' List 2.20
110 '
120 SCREEN 0,0 : CLS 3
130 RANDOMIZE VAL(RIGHT$(TIME$,2))
140 G=INT(RND*300)+250
150 LINE (G-2,0)-(G+2,199),6,BF
160 '
170 X=0 : Y=100 : C=5
180 GOSUB *MAN :  BEEP 1 : BEEP 0
190   IF X+36>G THEN BEEP : END
200   X=X+30
210 GOTO 180
220 '
230 *MAN
240 CIRCLE (X+28,Y+6),4,C
250 LINE (X+26,Y+8) -(X+18,Y+16),C
260 LINE (X+12,Y+15)-(X+14,Y+10),C : LINE -(X+24,Y+10),C
270 LINE            -(X+25,Y+14),C : LINE -(X+34,Y+13),C
280 LINE (X+18,Y+16)-(X+16,Y+21),C : LINE -(X+6,Y+24),C
290 LINE            -(X+10,Y+26),C
300 LINE (X+18,Y+16)-(X+28,Y+16),C
310 LINE            -(X+20,Y+22),C : LINE -(X+24,Y+22),C
320 RETURN
```

List 2.20a

```
190   IF X+6>G THEN BEEP : END
```

ルインと判定する」という場合には，190 行をどのように変えればいいでしょう？ はいそうですね．*List 2.20a*のように，IF～THEN 命令の判断を変えればよいのです．

GET & PUT

同じ形の絵や図形を１つの画面に描くために，とても便利な命令があります．GET 命令と PUT 命令がそれです．ここではこの２つの命令を使って動きを持ったグラフィックの表示を試みてみます．

それでは *List 2.21* のプログラムを実行してみてください(*Figure 38* 参照)．落下傘が開いてユラユラ降りていくでしょう．下方に位置する水色の着地台にみごと着地すればおなぐさみ．さて結果は？

140～220 行で，画面の上の方に１つ，落下傘が描かれます．ここまでは今までと別段たいした違いはありません．つまり LINE 命令や CIRCLE 命令を駆使したものです．

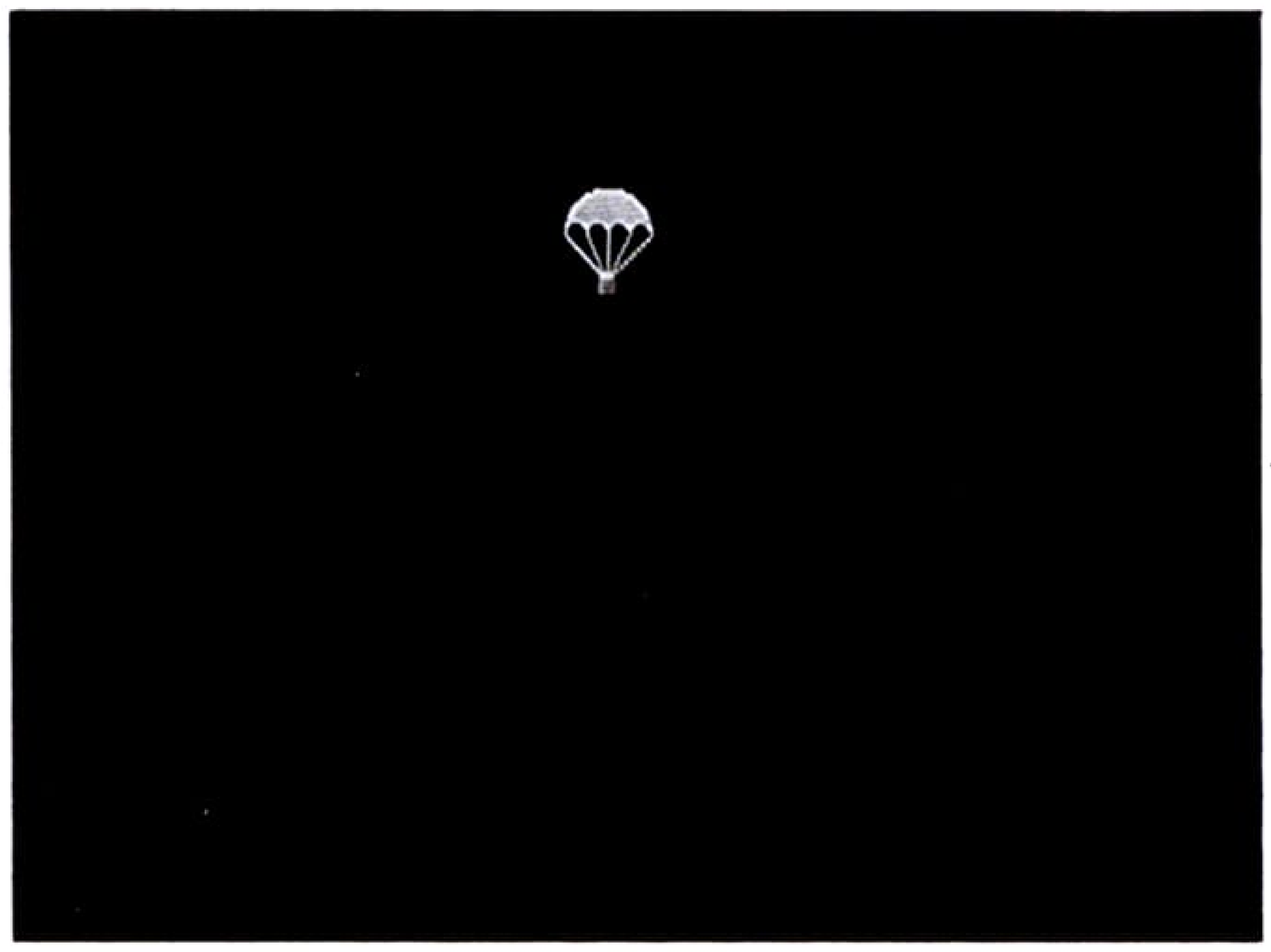


Figure 38

List 2.21

```
100 ' List 2.21
110 '
120 SCREEN 0,0 : CLS 3
130 DIM A%(288)
140 CIRCLE (333,11),20,7,0,3.14
150 FOR X=318 TO 348 STEP 10
160   CIRCLE (X,11),5,7,0,3.14
170 NEXT X
180 FOR X=313 TO 353 STEP 10
190   LINE (X,11)-(333,21),7
200 NEXT X
210 PAINT (333,3),5,7
220 LINE (329,21)-(337,24),6,BF
230 X=310 : Y=0
240 GET (X,Y)-(X+45,Y+24),A%
250 '
260 LINE (310,197)-(318,199),3,BF
270 *LOOP
280   IF Y>171 THEN BEEP : END
290   X=X+INT(RND*5)-2 : Y=Y+1
300   PUT (X,Y),A%,PSET
310 GOTO *LOOP
```

その次が重要.240行では,画面に描かれた絵のデータを,GET命令を使って配列A%に取り込みます.これで準備OK! PUT命令を使い,今取り込んだ絵を,どこにでも表示できるようにしたのです.

270〜310行のループで落下傘が降りていく様子を表示しています.なお,DISK BASICを使っている人は,ちょっと目先を変えてWHILE〜WENDでループを構成してみるとおもしろいでしょう(*List 2.21a* 参照).

さて,話を *List 2.21* に戻します.290行では,まず落下傘の位置のx座標に−2〜+2の乱数を加え,左右の位置を揺らします.そして次にy座標に1を足し,1ドット分降下させます.そうして得られた新しい位置に,300行のPUT命令で落下傘を表示します.

あれ,前に描いた絵は消さなくていいの?という疑問をお持ちの方もいるでしょう.前に描いた絵を次々に消しながら新しい絵を表示しなくては,動いた跡が残ってしまうはずですよね.実はこのPUT命令が,新しい位置へ絵を表示すると同時に,古い絵の残骸を消す仕事も行っているのです.

秘密はGET命令実行時の座標の指定にあります.絵を配列に取り込むときに,落下傘の絵が存在する範囲ぎりぎりに座標の指定をしないで,左右に2ドットずつ,そして上に1ドット,余分な空白部分をみこんで座標を指定したのです.

PUT命令を実行するとき,新しい座標は古い座標に比べて,下方に1ドット移動し,左右はせいぜい2ドット分しか揺れないのですから,このGET命令で読み込んだ余分の空白部分が,ちょうど前に描いた絵の残骸を消す役目を果たしてくれるのです.

もしこのような工夫をせず,落下傘のみを配列に取り込んだらどうなるか,試してみましょう.230,240行を *List 2.21b* のように変更してください.

前に描いた落下傘の一部が消されずに,跡を残してしまいます.

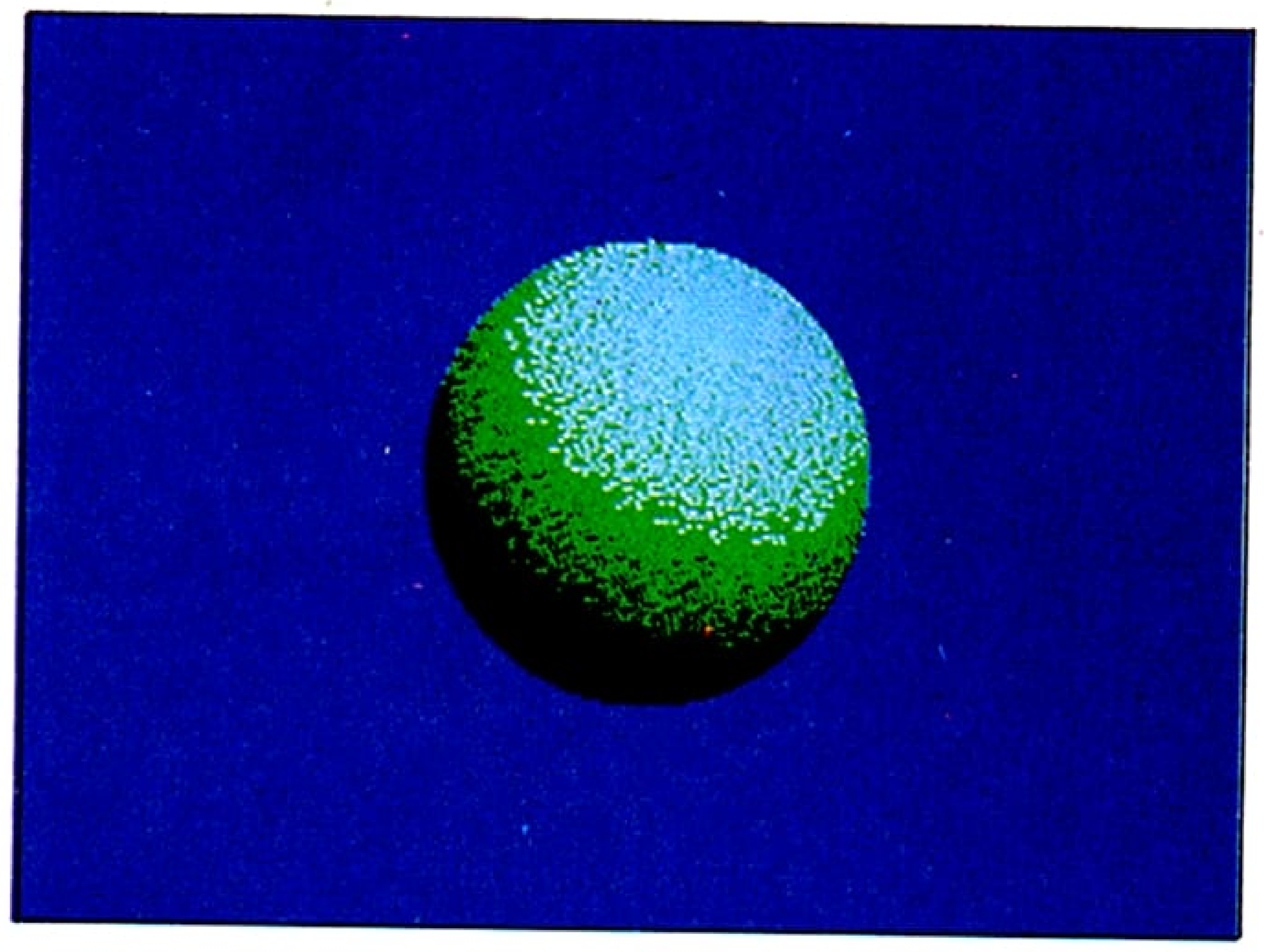
List 2.21a

```
270 WHILE Y<172
280   X=X+INT(RND*5)-2 : Y=Y+1
290   PUT (X,Y),A%,PSET
300 WEND
310 BEEP : END
```

List 2.21b

```
230 X=312 : Y=1
240 GET (X,Y)-(X+41,Y+23),A%
```

暑さ寒さはコサインで



気候というものは土地によって様々です．灼熱の太陽照りつける熱帯の国々がある一方で，南極や北極という氷に閉ざされた厳寒の地もまた，現に存在しているのです．同じ地球の上にあって，なぜこのような違いが生ずるのでしょうか．

この気候を左右するのは，太陽までの距離ではなく，実は光線の入射する角度が一番の要因なのです．太陽が頭の真上から照らせば気温は高くなり，太陽の位置が低ければ気温も低くなります．

太陽光線の入射方向と垂直との角度を θ とすれば，地面が受ける入射エネルギーの量は $\cos\theta$ に比例します．

そこで，いま明らかにされた暑さ寒さと $\cos\theta$ の意外な関係,これをグラフィックで表示しようと作成したのが，何をかくそう下記のプログラム．実行開始から約25分，コーヒーでも飲みながらお待ちください．右上方向からの光線に照らされた球体が次第に姿を現します．

```
100 ' List Ex.2
110 '
120 CONSOLE,,0 : CLS
130 SCREEN 0,0 : LINE (0,0)-(639,199),1,BF
140 '
150 R=100 : UX=1 : UY=2 : UZ=2
160 '
170 P=.5
180 U=SQR(UX*UX+UY*UY+UZ*UZ) : UX=UX/U : UY=UY/U : UZ=UZ/U
190 C(0)=0 : C(1)=4 : C(2)=7
200
210 FOR Y=-R TO R STEP 1/P
220   FOR X=-R TO R
230     Z2=R*R-X*X-Y*Y
240     IF Z2<0 GOTO 290
250       Z=SQR(Z2)
260       CS=(X*UX+Y*UY+Z*UZ)/R
270       E=INT(CS*2+RND) : IF E<0 THEN E=0
280       PSET (320+X,100-Y*P),C(E)
290   NEXT X
300 NEXT Y
```

【3章】

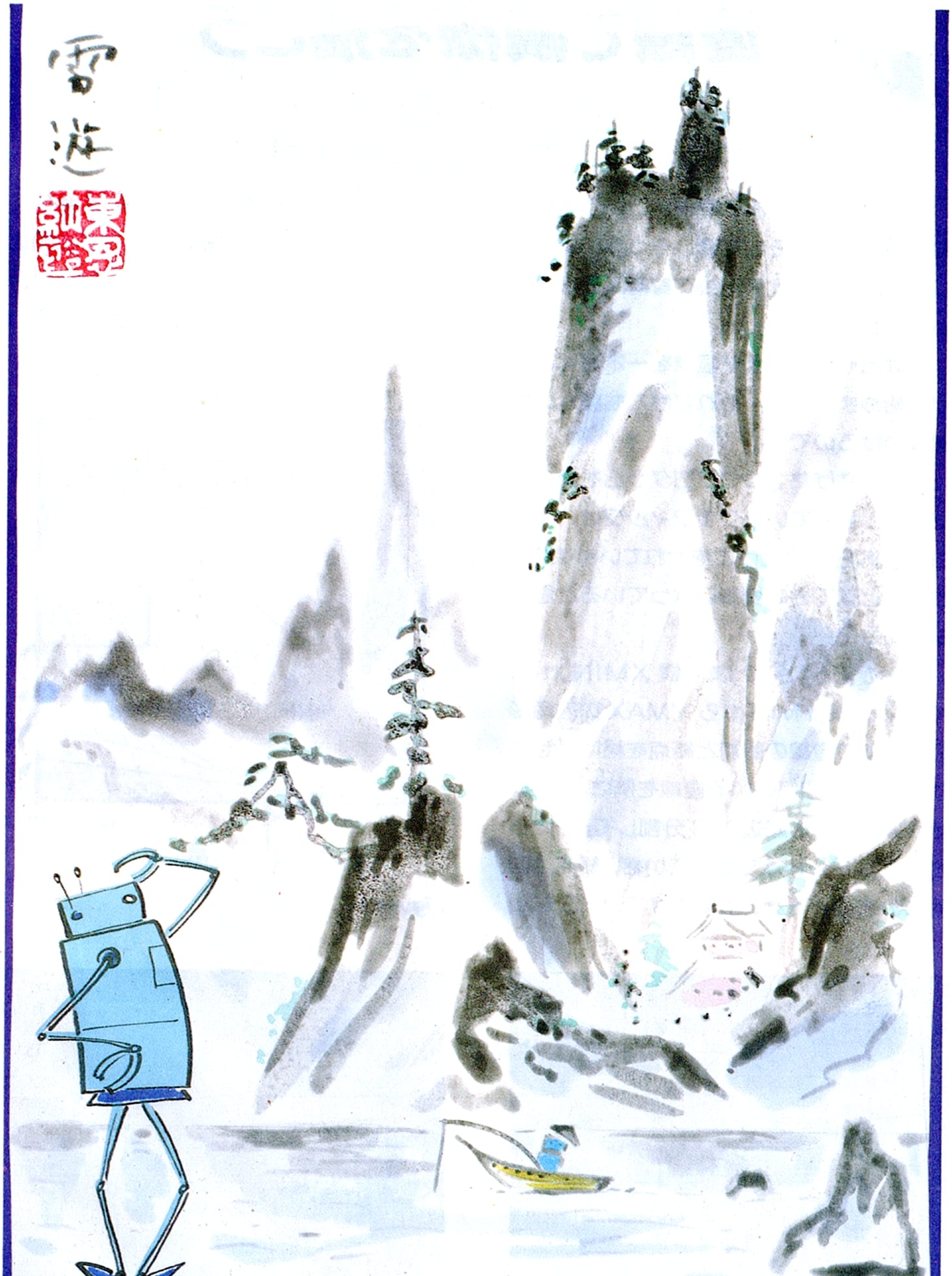
アートへのアプローチ



直線で模様を描こう
MUG CUPS
リンゴを描く
落葉と星型
Stardust Night
円は異なもの味なもの
自分だけの曲線を描こう
ヘリコイド曲線を描く
ネコでも回そうか?
3次元グラフィックス?
クリスマス・ツリー

本章ではコンピュータによるアートに挑戦します.

近年,我等がPC-8801 mk IIをはじめとする高性能パソコンが出現し,これまでは高嶺の花であったコンピュータ・グラフィックスが私たちにも手軽に楽しめるようになりました.なかでも,本章に掲載したような世人の目を楽しませる壮麗な,あるいは小意気に洒落たグラフィック作品が,わずかな手間で描けるこの楽しさ…….美しい作品を描くためのテクニックのその入口あたりを種々取りまぜて見てもらうことにいたします.月並ないい方ですが,これを基礎として皆さん自身でいろいろなグラフィックに取り組んでいただきたい,筆者らはそう希望しております.


雪遊

直線で模様を描こう

本章のテーマであるアートという言葉に何やら高尚な，そして難しそうな印象を感じた方，いえいえそんなに身構える必要はありません.

何はともあれ，まず直線を一本引くことから始めましょう．それだけでも美しい模様は描けるのです.

ここに挙げた 2 つのプログラムを見てください．扱っているグラフィックは，どちらもLINE 命令だけで描かれていますが，ちょっとすてきな模様になっていると思いませんか?

List 3.1a では，横 XMIN から XMAX，縦 YMIN から YMAX の四角形の各辺上で直線の始点と終点を移動させます．190 行で引いている直線を例にとれば，始点は四角形の底辺を S 分割して，左から右に XS ドットずつ動いていき，終点は四角形の右辺を下から上に向けて，YS ドットずつ動いていきます（*Figure 1* 参照）．この始点と終点を順に結ぶと，右下の角に相当する部分が描きあがります．同様に，200～220 行までの LINE 命令が，四角形の

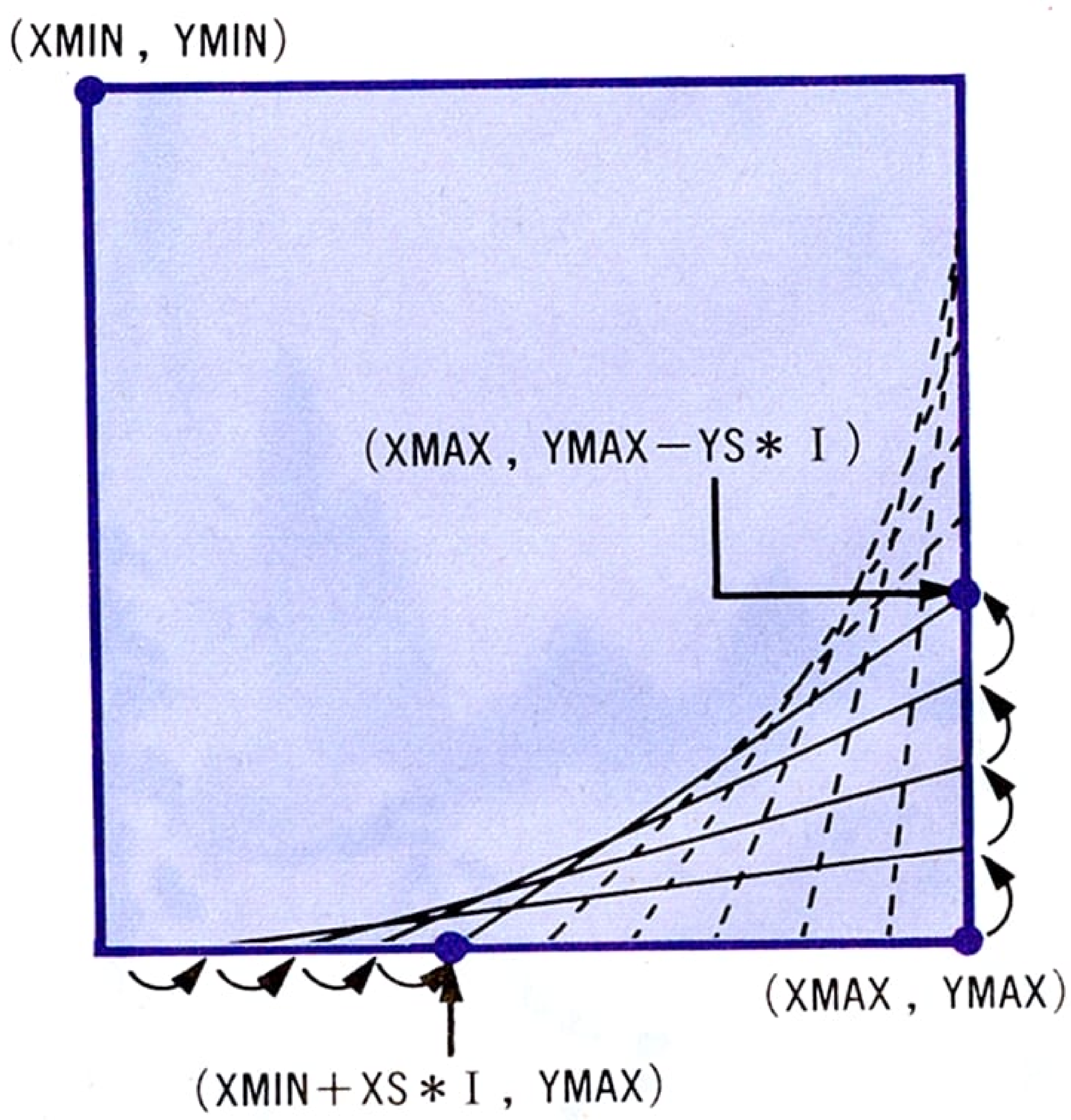
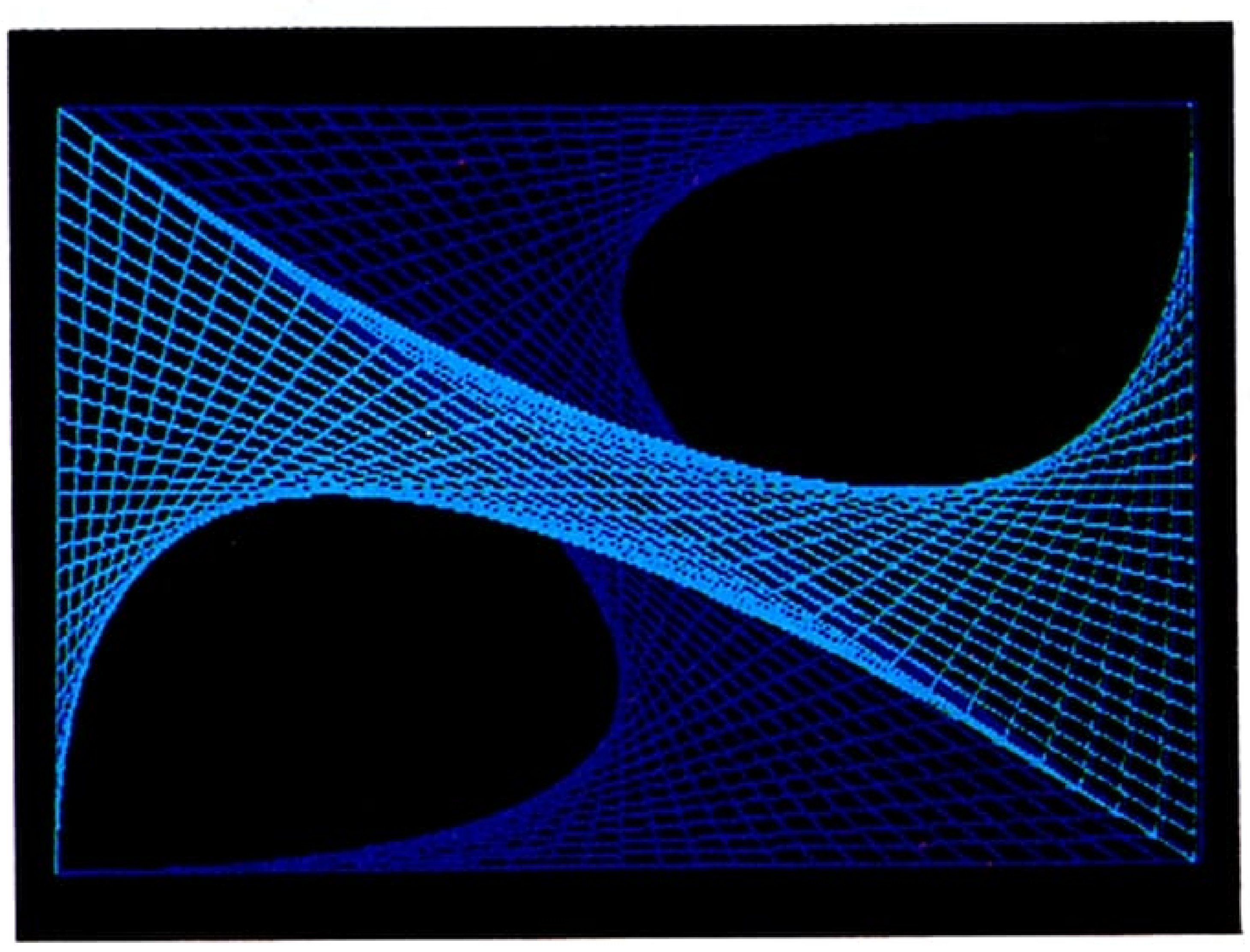
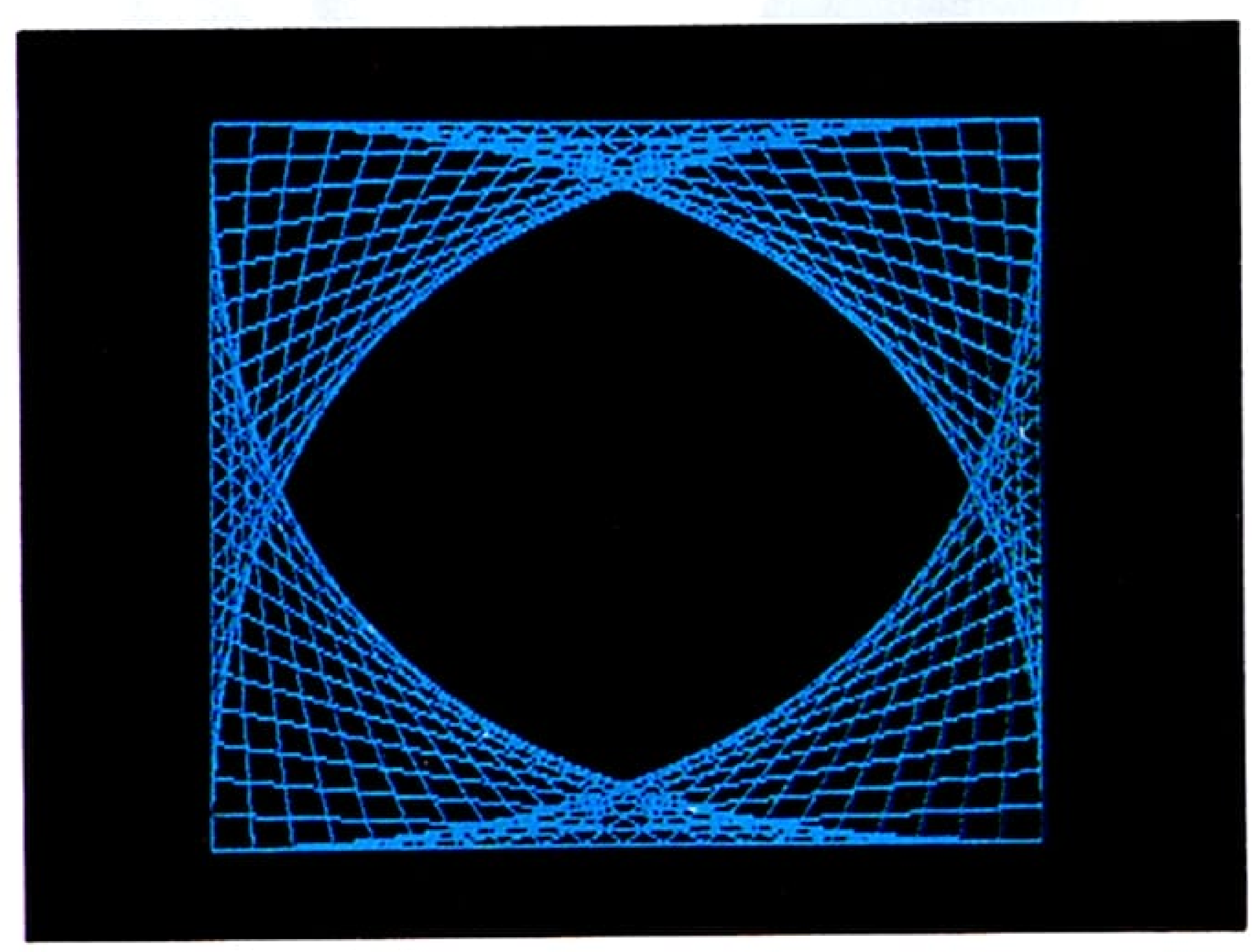


Figure 1





右上，左上，左下の角を描いていくわけです．

List 3.1b でも，やはり直線の始点と終点を動かしていますが，今度は四角形の辺と対角線を結んで線を引いています．

どちらのプログラムも変数 S や XMAX，YMAX などの値を変えると画面が変わります．変化する模様を楽しんでください．

List 3.1a

```
100 ' List 3.1a
110 '
120 XMIN=100  : YMIN=10
130 XMAX=540  : YMAX=190
140 S=20      : C=5
150 XS=(XMAX-XMIN)/S : YS=(YMAX-YMIN)/S
160 '
170 SCREEN 0,0 : CLS 3
180 FOR I=0 TO S
190   LINE (XMIN+XS*I,YMAX)-(XMAX,YMAX-YS*I),C
200   LINE (XMAX-XS*I,YMIN)-(XMAX,YMAX-YS*I),C
210   LINE (XMAX-XS*I,YMIN)-(XMIN,YMIN+YS*I),C
220   LINE (XMIN+XS*I,YMAX)-(XMIN,YMIN+YS*I),C
230 NEXT I
```

120～130：グラフィックを表示する範囲
140：各辺を分割する数と色の設定

List 3.1b

```
100 ' List 3.1b
110 '
120 XMIN=0    : YMIN=0
130 XMAX=639  : YMAX=199
140 C1=1      : C2=5
150 S=30
160 XS=(XMAX-XMIN)/S : YS=(YMAX-YMIN)/S
170 '
180 SCREEN 0,0 : CLS 3
190 FOR I=0 TO S
200   LINE (XMAX-XS*I,YMIN)-(XMIN+XS*I,YMIN+YS*I),C1
210   LINE (XMIN+XS*I,YMAX)-(XMAX-XS*I,YMAX-YS*I),C1
220 NEXT I
230 FOR I=0 TO S
240   LINE (XMAX,YMIN+YS*I)-(XMAX-XS*I,YMAX-YS*I),C2
250   LINE (XMIN,YMAX-YS*I)-(XMIN+XS*I,YMIN+YS*I),C2
260 NEXT I
```

140：色の設定

MUG CUPS

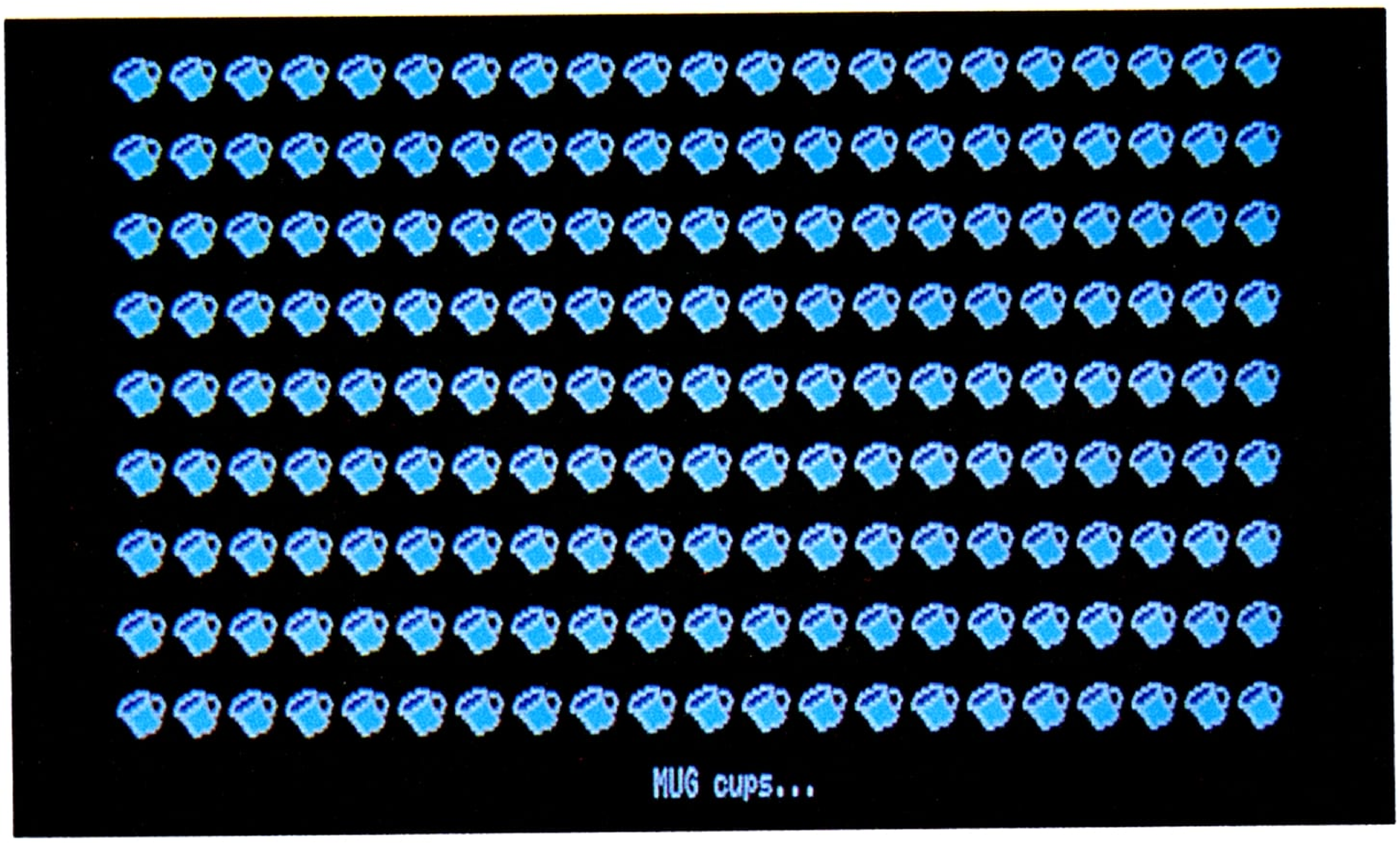
1つの図形を繰り返し描くことによって作り出される模様は，布地や壁紙などに数多く見受けられます．ここでは，かわいいマグカップを繰り返し描き，模様を作ってみましょう．

このプログラムは，基本的には，180～220行の2重のFOR～NEXTループが，マグカップを描く始点の位置を決めています．そして，270行以降のサブルーチンが，その始点をもとに，ひとつのマグカップを描きます．

360行以降のデータは，（0，0）を原点と考えたときの図形の各点の座標を示しています．なお，最後のデータ（−1，−1）は座標ではなく，データの終わりを示すために任意に決めた数で，この数はデータの中に出てきそうもない数であれば，何を使ってもかまいません．

280行では，マグカップを描き出すための始めの点を打ち，310行のLINE命令が次々と連続して直線を引いていき，マグカップを完成させます．ちゃんと原点(X，Y)に，読み取ったデータA，Bを足して出力していますね．

AとBがともに−1になったら，その時点でデータが最後だと判断し，線をのばす



のをやめます．このような形でプログラムを組んでおくと，データを変更してもその個数を数え直す必要がありません．

240 行では，何かキーが押されるまで次のプログラムが実行されないようにしています．250 行の END 命令が実行され，画面をスクロールし，OK を表示しないようにしているのです．

このプログラムに使われているマグカップのデータは，Appendix に掲載してあるデザインシートを使って，鉛筆で描いた下絵から読み取りました．デザインシートは PC-8801 mk II 用にマス目の縦横比が 1：0.5 になっていますが，もし普通の方眼紙を使うならば，読み取った縦の長さには 0.5 を掛けておく必要があります．

このように，コンピュータ・グラフィックスで望みの形を作り出す場合には，まずデザインシートに描いてみるという，基本的な作業をめんどくさがってはいけません．遠回りのように思えても，これが一番確実です．

List 3.2

```
100 ' List 3.2
110 '
120 XMIN=0    : YMIN=0          } グラフィックを表示する範囲
130 XMAX=620  : YMAX=170        }
140 XS=30     : YS=20           }
150 '
160 CONSOLE 0,25,0,1 : WIDTH 80,25              }
170 SCREEN 0,0 : CLS 3                          }
180 FOR Y=YMIN TO YMAX STEP YS                  }
190   FOR X=XMIN TO XMAX STEP XS                }
200     GOSUB 270                               } メインルーチン
210   NEXT X                                    }
220 NEXT Y                                      }
230 LOCATE 36,23 : PRINT "MUG cups..."          }
240 IF INKEY$="" GOTO 240                       }
250 END                                         }
260 '
270 RESTORE 360                                                   }
280 READ A,B : POINT (X+A,Y+B)……描き出すための処理                 }
290 READ A,B                                                      }
300  IF A=-1 AND B=-1 GOTO 330……データが終わりかどうかの判定       } サブルーチン
310   LINE -(X+A,Y+B),7……線をつなぎ，のばす                       }
320 GOTO 290                                                      }
330 PAINT (X+20,Y+10),5,7 : PAINT (X+14,Y+7),1,7
340 RETURN
350 '
360 DATA  5, 9, 10, 9, 19, 6, 20, 4, 16, 4,  7, 6,  5, 8
370 DATA  9,10, 15,14, 22,13, 26,11, 21, 7, 22, 5, 26, 5
380 DATA 28, 7, 27, 9, 23, 9, 21, 7, 20, 4, -1,-1
```

リンゴを描く

MUG CUPS のプログラムでは，一筆書きでできる図形のデータを DATA として蓄えておく方法を示しました．それでは，何本もの線で構成されている図形を描くにはどうしたらよいのでしょうか．ここに一例を示しました．

170～220 行の 2 重ループで図形の原点を決めているのは，MUG CUPS のプログラムと同様です．250 行以降が，リンゴを描くサブルーチンです．

サブルーチン内で使われているデータは，やはり，座標（0，0）を原点とする相対的なものです．しかし，その数値は文字変数として読み取っています．なぜなら，データ内の区切りに“end”，“p”，“／”のマークを使うからです．ここでは“p”，“／”のマークを線の切れ目として，“end”のマークをデータの終わりとして使っています．

その他のデータは 3 つ 1 組で，順に，x 軸方向の座標，y 軸方向の座標，直線の色，となっています．データは読み取った後に，VAL 関数で文字データから数値データに変換して利用します．

250 行や，それ以降で見られる FLAG という変数は，1 つ前に引いた直線が，次に引く直線とつながるものかどうかを示す目印です．FLAG の値は，A$が“p”を読み込んだときと“／”を読み込んだときだけ −1 となります．

もし FLAG＝−1 ならば，その 1 つ前の直線と，今読み込んだ座標（X，Y）は，まったくつながりを持たず，次の直線の始点となるわけです．310 行の，

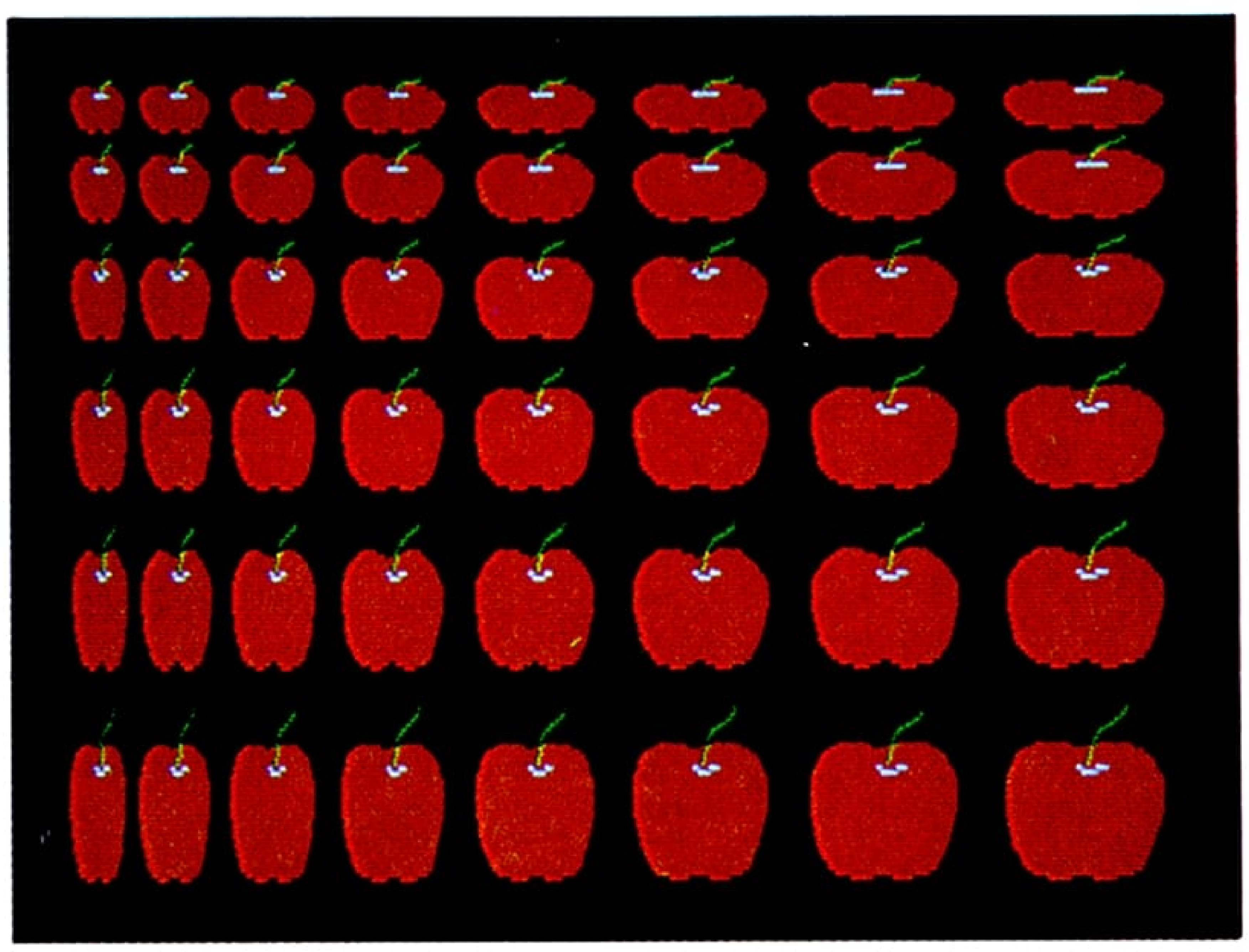
```
POINT (X,Y)
```

という命令で，その始点を画面上に決めています．

ここで始点が決まったら，次の“p”と“／”のデータを読み，FLAG＝−1 となるまで，直線をつないでいくわけですから，FLAG は 0 にしておかなければなりません．

FLAG=0 ならば，

```
LINE -(X,Y),C
```



の命令で，次々と点をつないでいくことになります．320行でこの処理を行っています．

同じリンゴをずらずら並べても，おもしろくないので，190行のVIEW命令でリンゴを変形してみました．図形の原点（I，J）をVIEW命令の終点座標に加えて，右に表示されるリンゴほど横方向へ拡大し，下に表示されるリンゴほど縦方向へ拡大しています．

List 3.3

```
100 ' List 3.3
110 '
120 XMIN=100 : YMIN=20
130 XMAX=539 : YMAX=159
140 XS=55    : YS=25
150 '
160 SCREEN 0,0 : CLS 3 : WINDOW (0,0)-(639,199)
170 FOR J=YMIN TO YMAX STEP YS
180   FOR I=XMIN TO XMAX STEP XS
190     VIEW (0,0)-(I+100,J+40)
200     GOSUB *APPLE
210   NEXT I
220 NEXT J
230 VIEW (0,0)-(639,199) : END
240 '
250 *APPLE : FLAG=-1 : RESTORE 360
260 READ A$
270   IF A$="end" THEN 340
280   IF A$="p" THEN FLAG=-1 : PAINT (I+30,J+30),C : GOTO 260
290   IF A$="/" THEN FLAG=-1 : GOTO 260
300   READ B$,C$ : X=VAL(A$) : Y=VAL(B$) : C=VAL(C$)
310   IF FLAG=-1 THEN POINT (X+I,Y+J) : FLAG=0
320   LINE -(X+I,Y+J),C
330 GOTO 260
340 RETURN
350 '
360 DATA 56,19,2, 69,17,2, 78,17,2, 89,20,2, 96,25,2, 96,32,2
370 DATA 91,40,2, 85,45,2, 78,47,2, 67,48,2, 56,46,2, 44,48,2
380 DATA 33,47,2, 27,45,2, 20,40,2, 16,32,2, 16,25,2, 22,20,2
390 DATA 33,17,2, 42,17,2, 56,19,2,p
400 DATA 76, 9,4, 76,10,4, 71,12,4, 62,14,4, 58,18,4, 56,22,4,/
410 DATA 51,23,7, 56,24,7, 65,23,7,end
```

300行の注：……文字データを数値に変換する

360行の注：1組のデータ（56,19,2）；89,20,2 の各値 → x座標，y座標，線の色

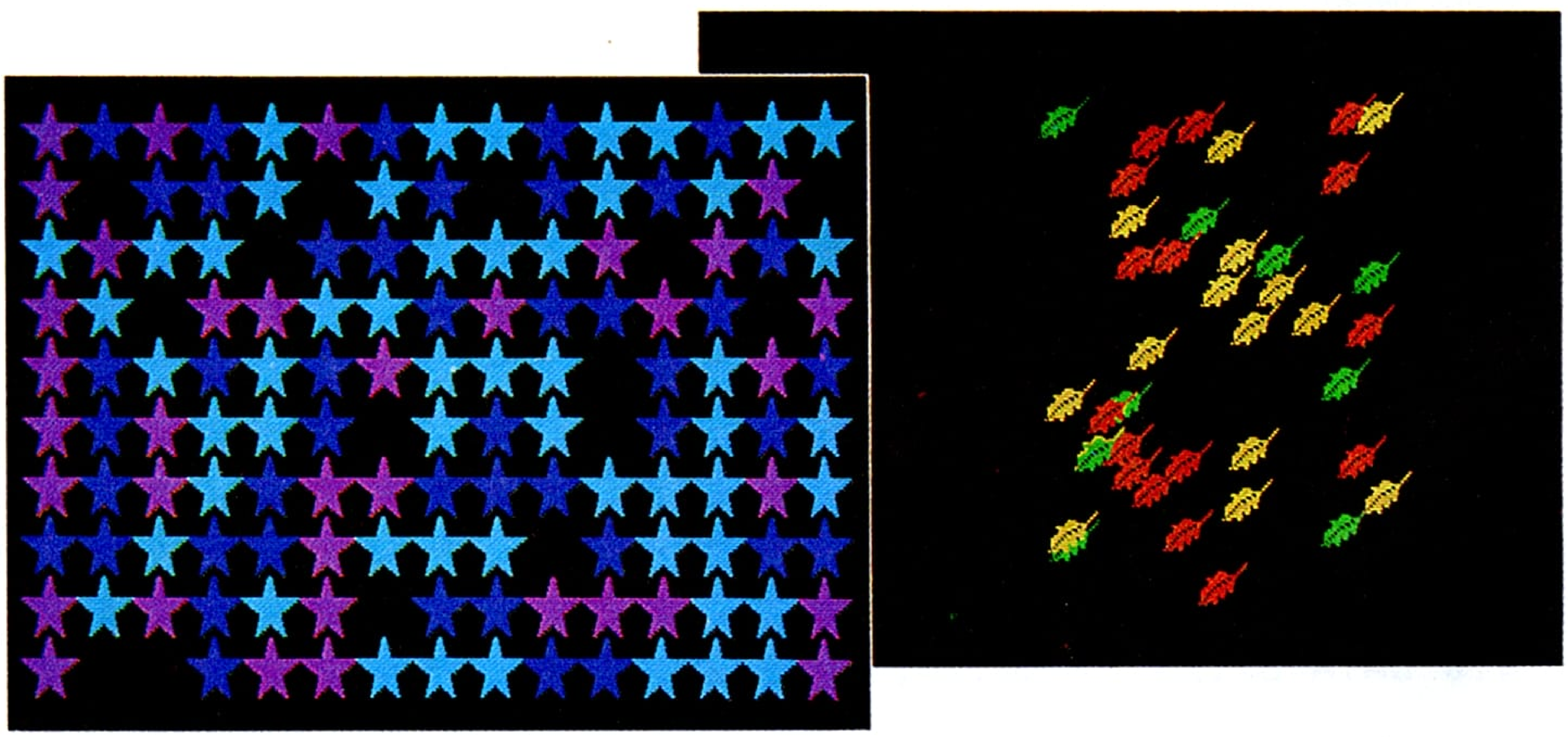
落葉と星型

RNDという関数をグラフィックに利用してみましょう．いうまでもありませんが，乱数は何の一定の規則的な並びを持たない数列です．それを利用したのが次のプログラムです．

List 3.4a は，落葉を描くプログラムです．240行以降のサブルーチンは，（X，Y）を原点として，落葉の形を出力するためのものです．この原点（X，Y）は乱数で決めています．180行では，XMINより大きく，XMIN+XWより小さい数になるような乱数を発生させて，変数Xに代入しています．同様にして，YMINより大きく，YMIN+YWより小さくなるような乱数を発生させ，変数Yに代入しています．190行では，2，4，6をランダムに発生させ，変数Cに代入して落葉の色を決定しています．

List 3.4b は，星型を描くプログラムです．*List 3.4a* と同じ考え方で，220行以降は，星型を画面に出力するサブルーチンになっています．160行で，変数Xには，0からXD＊XSまでの数をXSおきに発生させ，変数Yには，0からYD＊YSまでの数をYSおきに発生させて代入しています．つまり，得られる数が，おのおのXS，YSおきの乱数となっており，そのため星型が重なり合うことはありません．

この *List 3.4b* ではXS，YSの値を小さくしてみて，画面がどのようになるか試してみてください．



List 3.4a

```
100 ' List 3.4a
110 '
120 XMIN=200 : YMIN=20
130 XMAX=440 : YMAX=180
140 XW=XMAX-XMIN : YW=YMAX-YMIN……発生させる乱数の範囲を設定
150 '
160 SCREEN 0,0 : CLS 3
170 FOR I=1 TO 40
180   X=INT(RND*XW)+XMIN : Y=INT(RND*YW)+YMIN ⎫
190   C=INT(RND*3)*2+2                        ⎭ 乱数による数値の発生
200   GOSUB *LEAF
210 NEXT I
220 END
230 '
240 *LEAF
250 COLOR ,,C : LINE (X+40,Y)-(X+32,Y+4)
260 LINE -(X+20,Y+3)  : LINE -(X+22,Y+4)  : LINE -(X+12,Y+5)
270 LINE -(X+16,Y+6)  : LINE -(X+8,Y+9)   : LINE -(X+8,Y+12)
280 LINE -(X+6,Y+13)  : LINE -(X+16,Y+12) : LINE -(X+18,Y+13)
290 LINE -(X+20,Y+10) : LINE -(X+24,Y+11) : LINE -(X+26,Y+8)
300 LINE -(X+30,Y+9)  : LINE -(X+32,Y+4)  : PAINT (X+18,Y+7)
310 LINE (X+29,Y+5)-(X+20,Y+7),0 : LINE -(X+7,Y+13),0
320 RETURN
```

List 3.4b

```
100 ' List 3.4b
110 '
120 XD=15 : YD=10 : XS=40 : YS=20
130 '
140 SCREEN 0,0 : CLS 3
150 FOR I=1 TO 400
160   X=INT(RND*XD)*XS : Y=INT(RND*YD)*YS ⎫
170   C=INT(RND*3)*2+1                    ⎭ 乱数による数値の発生
180   GOSUB *STAR
190 NEXT I
200 END
210 '
220 *STAR : COLOR ,,,C
230 POINT (X+20,Y)      : LINE -(X+16,Y+7)  : LINE -(X+1,Y+7)
240 LINE -(X+15,Y+12) : LINE -(X+11,Y+19) : LINE -(X+21,Y+15)
250 LINE -(X+31,Y+19) : LINE -(X+27,Y+12) : LINE -(X+41,Y+7)
260 LINE -(X+25,Y+7)  : LINE -(X+21,Y)    : PAINT (X+20,Y+5),C
270 RETURN
```

Stardust Night

PC-8801 mk IIのグラフィック画面は，640×200ドットの解像度があり，画面をキャンパスに見たてて，絵を描くことも十分可能です．スターダストナイトはグラフィックのサンプルプログラムとしてはスタンダードなものですが，ここではそれをちょっと工夫して，夜の海辺の情景を描くプログラムに挑戦してみましょう．

List 3.5 のプログラムを見てください．まず，輝く海は180〜230行で表示しています．190〜220行のループで左から順にドットを表示させますが，200行でRND＊RND＊RND＊30を用いて乱数を発生させ，Y座標を決めてしまいます．こうすると，ドットが水平線近くに偏って描かれ，光り輝く波の様子が表現できます．



次に，摩天楼のシルエットを描きます．270行で乱数を用いて，摩天楼のそれぞれのビルの高さと幅を決め，250〜290行のループで，左から順に描いていきます．低いビルを多く，高いビルを少なくするために，高さはRND＊RNDを利用しています．

310，320行で三日月を描きます．円の中心座標を（XM，YM）としたのは，360行で星が月と重ならないようにするための処理をしているからです．

最後に，340行以降で，星を描きます．20個の星のそれぞれの位置は，350行の乱数で決定されますが，どこの座標に星が描かれるかという値は，370行でDX(I)，DY(I)という配列に保存しておきます．星の位置が決まったら400〜420行で，順に星を描いていきます．410行では，1つ星を描くごとにキーボードを読んで，何かキーが押されていたらプログラムを終了するという処理を行っています．最後の450〜520行のサブルーチンは，I番目の星を描くものです．

20個描き終わったら，再び始めから描き直します．470行で星の色を乱数で決めていますから，＊STARのサブルーチンを1回実行するごとに星は色を変えて描き直されます．その結果として，星はまたたいて見えるのです．

List 3.5

※

```
100 ' List 3.5
110 '
120 XM=153 : YM=43
130 DIM DX(20),DY(20),D(20),C(2) : C(0)=3 : C(1)=5 : C(2)=6
140 RANDOMIZE VAL(RIGHT$(TIME$,2))
150 '
160 SCREEN 0,0 : CLS 3 : LINE (0,0)-(639,199),1,BF
170 '
180 FOR C=4 TO 6
190   FOR X=0 TO 639
200     Y=RND*RND*RND*30
210     PSET (X,Y+159),C
220   NEXT X
230 NEXT C
240 '
250 FOR I=0 TO 19
260   X=200+I*10
270   XW=10+RND*10 : YH=10+RND*RND*60
280   LINE(X,159)-(X+XW,159-YH),0,BF
290 NEXT I
300 '
310 CIRCLE (XM,YM),17,6        : PAINT (XM,YM),6
320 CIRCLE (XM-15,YM-5),17,1 : PAINT (XM-13,YM-3),1,1
330 '
340 FOR I=1 TO 20
350   X=RND*639 : Y=RND*100
360   IF X>XM-30 AND X<XM+30 AND Y>YM-15 AND Y<YM+15 GOTO 350
370   DX(I)=X : DY(I)=Y
380 NEXT I
390 '
400 FOR I=1 TO 20
410   IF INKEY$<>"" THEN END ELSE GOSUB *STAR
420 NEXT I
430 GOTO 400
440 '
450 *STAR
460 X=DX(I) : Y=DY(I)
470 C=C(INT(RND*3))
480 POINT (X,Y)           : LINE -STEP(-3,4),7 : LINE -STEP(-4,0),7
490 LINE -STEP(5,3),7 : LINE -STEP(-2,3),7 : LINE -STEP(4,-3),7
500 LINE -STEP(4,3),7 : LINE -STEP(-2,-3),7: LINE -STEP(5,-3),7
510 LINE -STEP(-4,0),7: LINE -STEP(-3,-4),7: PAINT (X,Y+3),C,7
520 RETURN
```

- 180–230: 輝く海を描く
- 250–290: 摩天楼を描く
- 310–320: 三日月を描く
- 340–380: 星を描く
- 400–430: 星をまたたかせ，何かキーが押されたら終了する
- 450–520: 星のデータ

※ このマークが付いているプログラムは任意のキーを押して終了してください

円は異なもの味なもの

円を基本とした模様を描く際ひっきりなしに登場するのが，例の sin や cos という三角関数です．

Figure 1 のような直角三角形において，角 T と向かい合う位置の辺の長さが sin(T)に相当し，角 T と直角を結ぶ辺の長さが cos(T)に相当する，この位置関係を三角形がどんな向きに傾いても見分ける眼力が必要です．

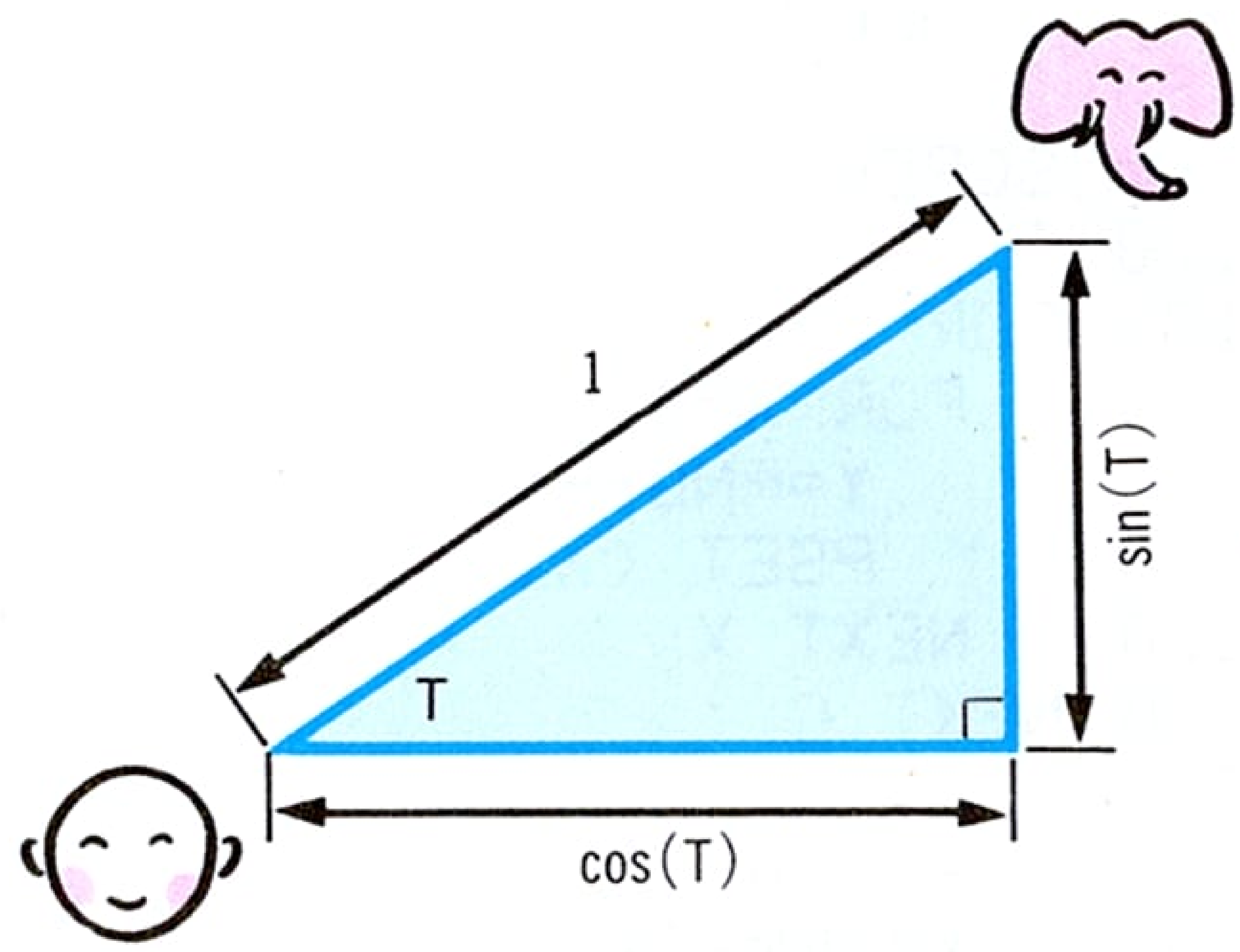
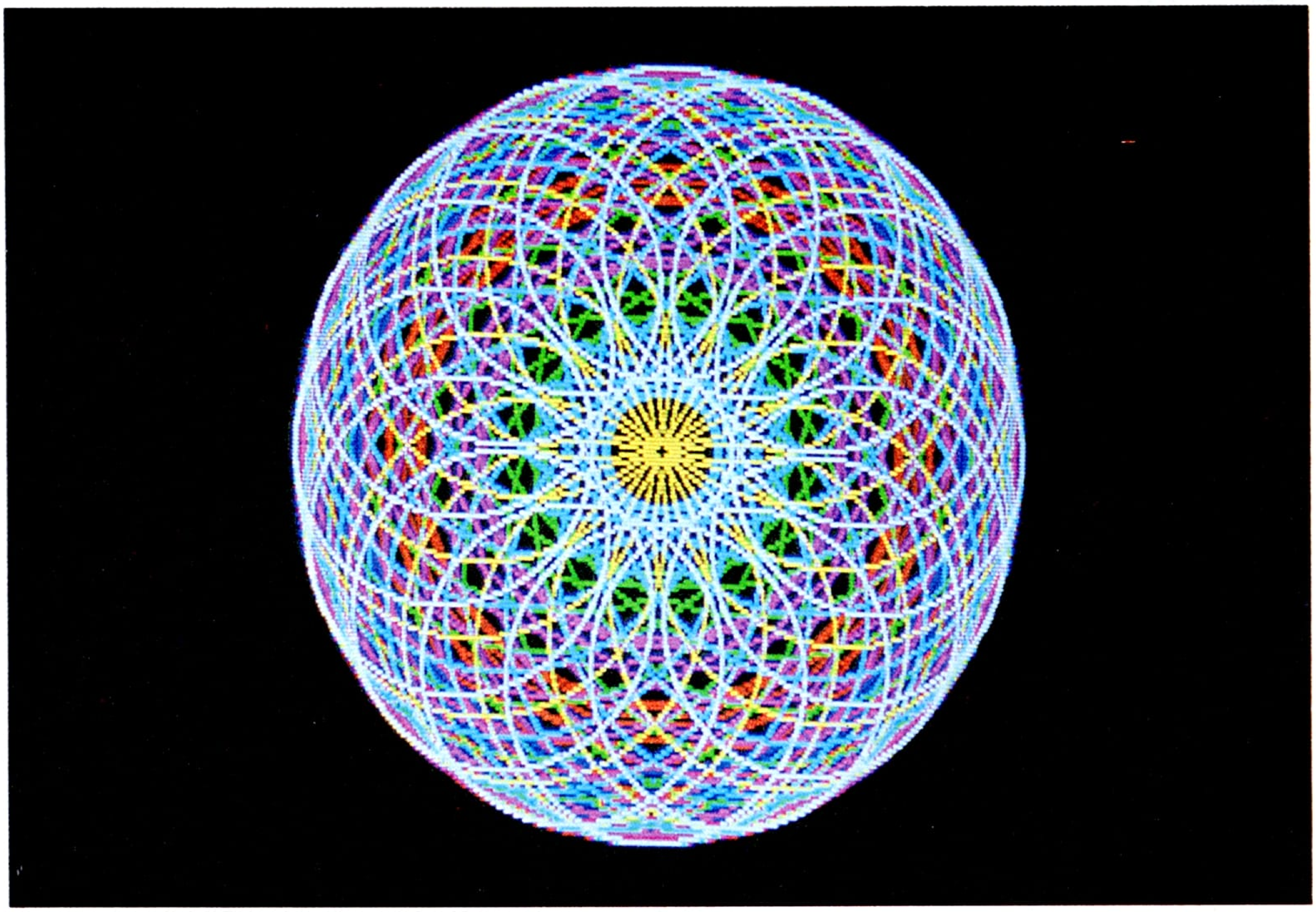


Figure 1



ここで *Figure 2* を見てください．始点から角度Ｔだけ回転した円周上の位置は，sin関数とcos関数を使って計算できます．この位置は，始点の位置と角度Ｔの向きによって様々な場合がありますが，ここでは2通りの例を示しました．どちらの場合も，画面上のｘ座標とｙ座標はそれぞれ右と下へいくほど増加するということと *Figure 1* の関係を考え合わせれば，答が出てきます．

円の中心を(Ｘ0，Ｙ0)，円の半径を１とすれば，*Figure 2a* のように，円の右端から反時計回りに角度Ｔだけ回転した位置は（Ｘ0＋cos(Ｔ)，Ｙ0－sin(Ｔ)），また，*Figure 2b* のように，円の下の端から時計回りに回転させる場合は（Ｘ0－sin(Ｔ)，Ｙ0＋cos（Ｔ)）というように位置の計算ができるわけです．

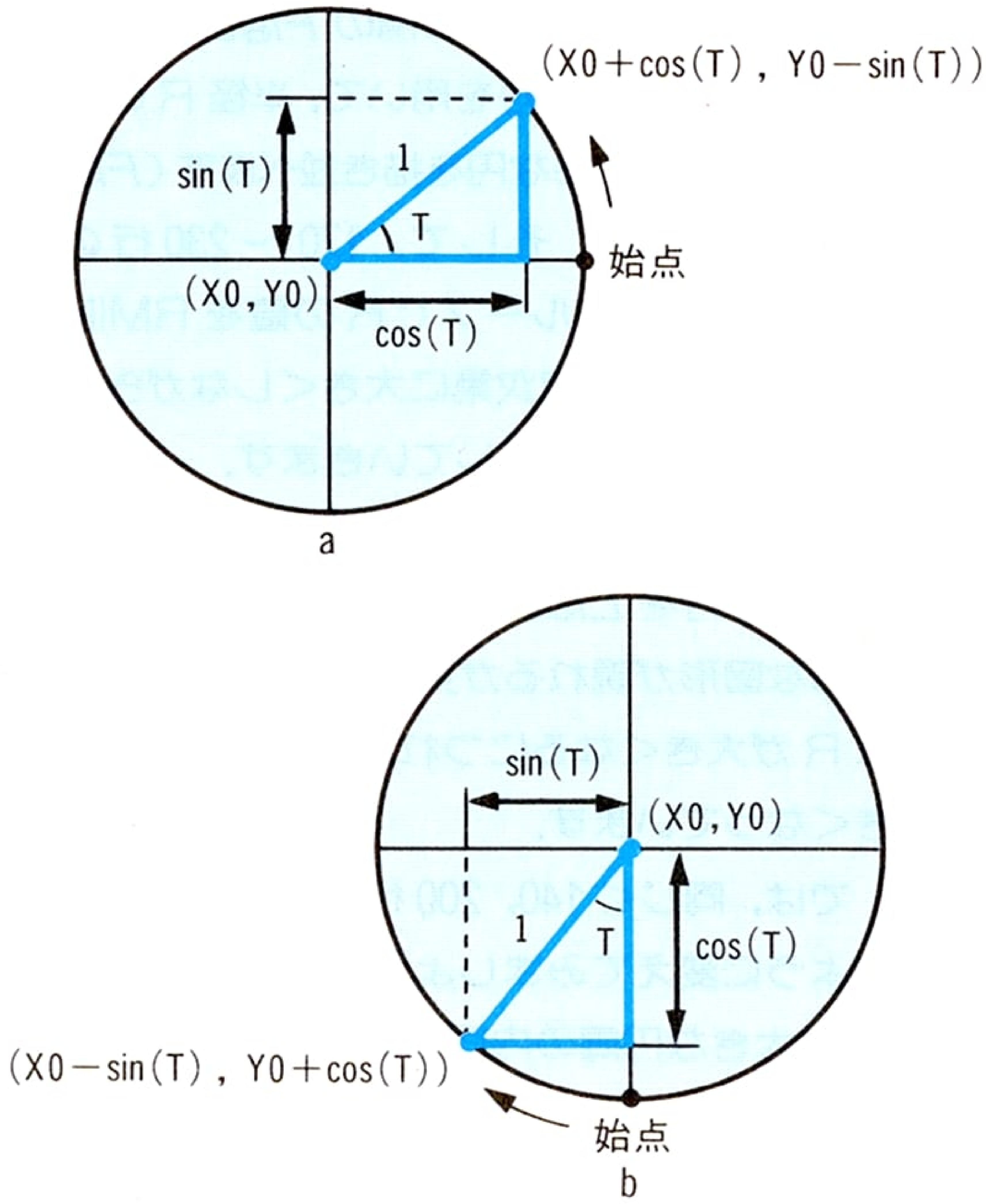
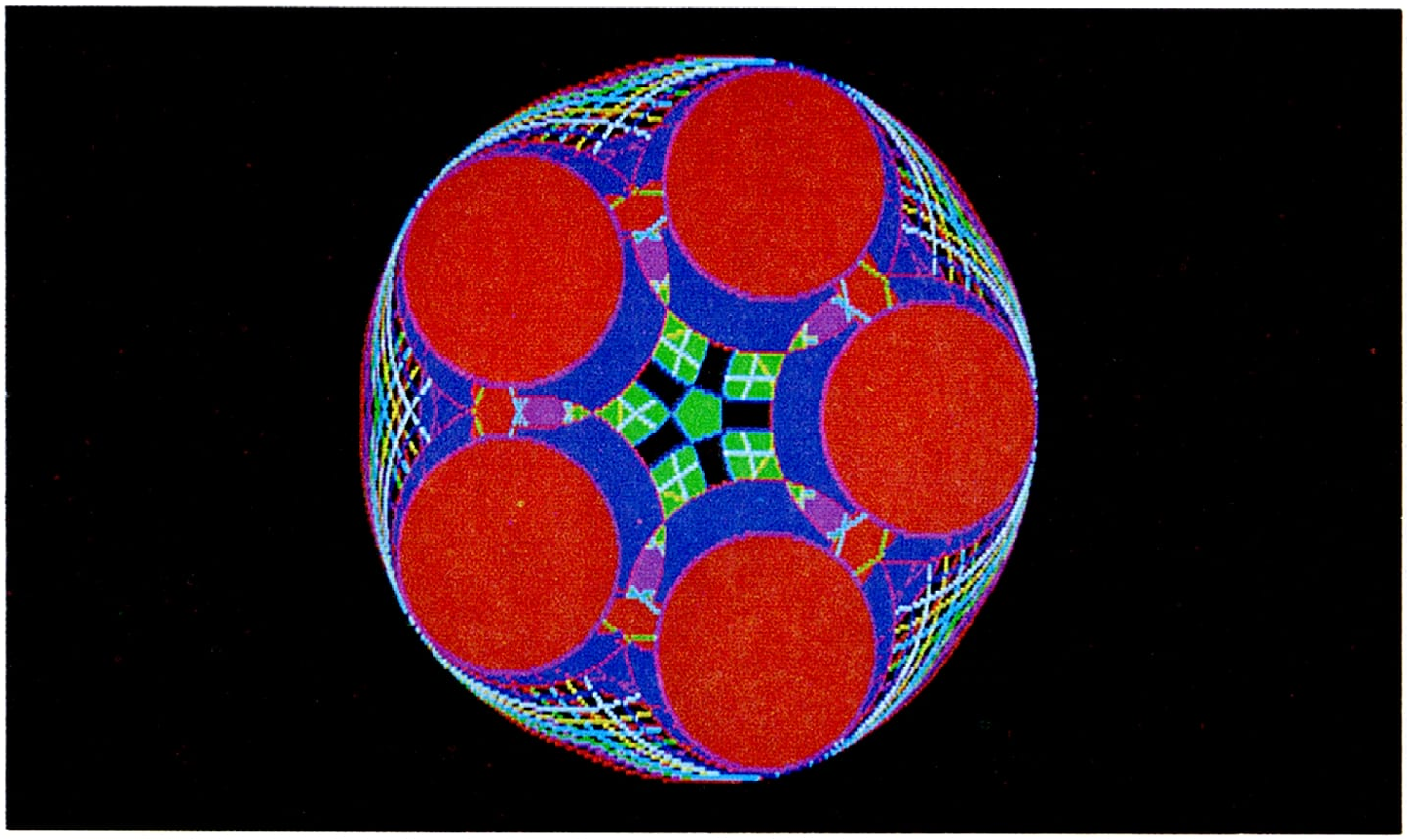


Figure 2



List 3.6 では，上の例の *Figure 2a* と同じ角度のとり方を用いて，半径 R の円周上に次々と小さな円を描き並べます（*Figure 3* 参照）．そして，170 ～ 230 行の FOR～NEXT ループで R の値を RMIN から RMAX まで次第に大きくしながら，同様なことを繰り返していきます．

さて，今のはほんの小手調べです．次に 140，200 行を *List 3.6a* のように変更し，どんな図形が現れるか見てください．これは R が大きくなるにつれ，描かれる円も大きくなっています．

では，同じく 140，200 行を *List 3.6b* のように変えてみましょう．今度は，半径 96 の大きな円周の内側に接して，円が並びます．

このように，円の半径の指定を変えるだけで，いろいろな変化が楽しめます．さらに，色を塗るときれいです．そこで，*List 3.6* に，*3.6c* や *3.6d* を追加してみてください．

どのプログラムも，140 行で色を決めています．円を並べるだけでも，こんなに目を楽しませてくれるグラフィックができあがるのです．

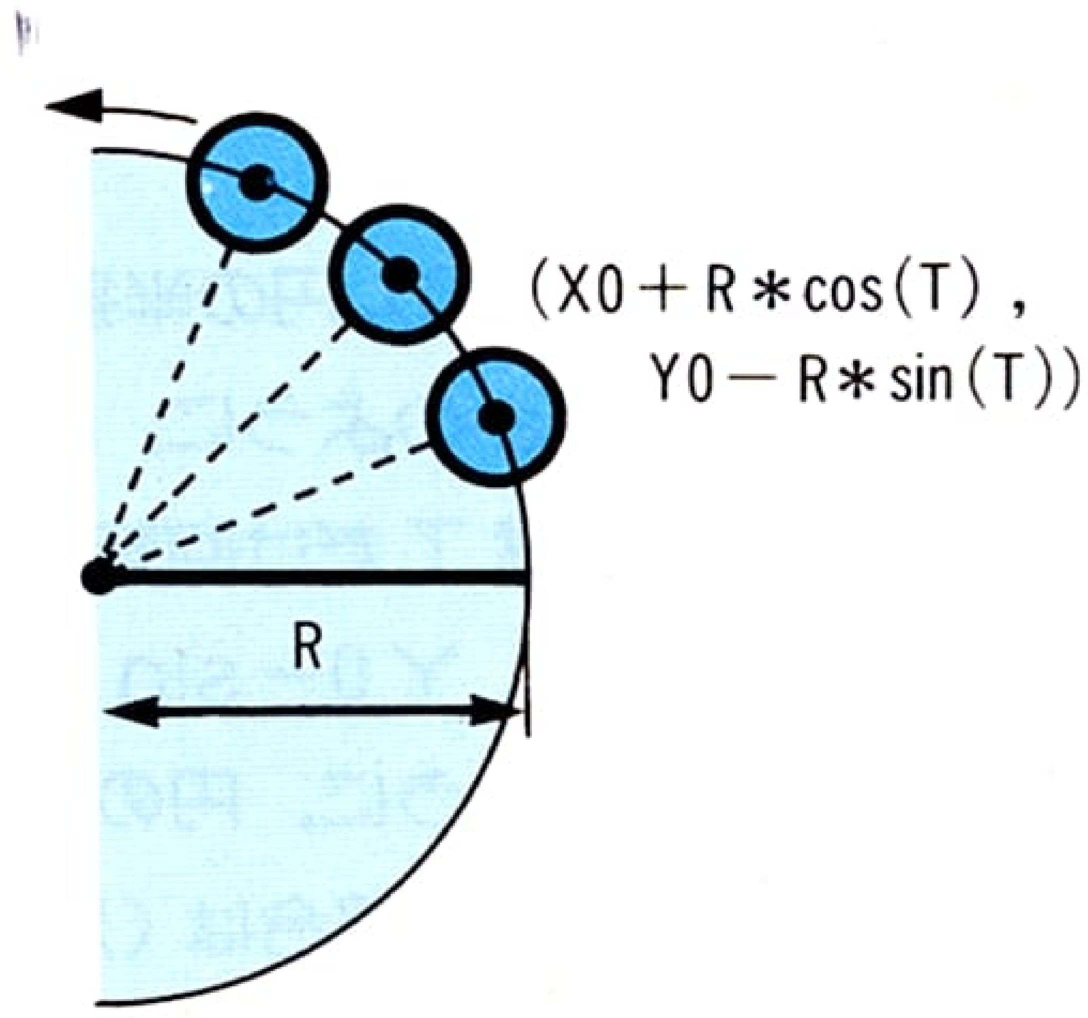
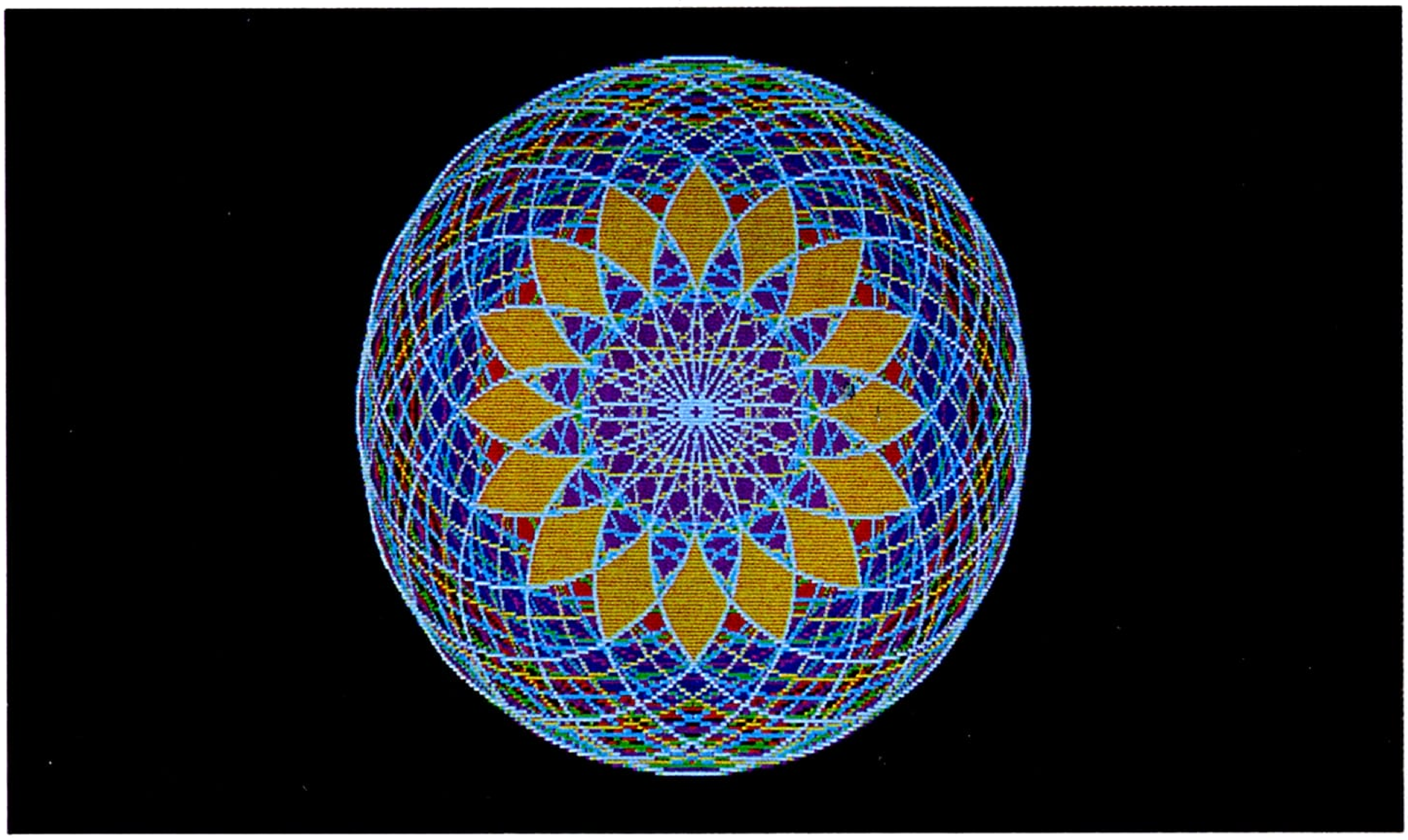


Figure 3



List 3.6

```
100 ' List 3.6
110 '
120 P=.5 : PI=3.14159……定数の設定
130 RMIN=18 : RMAX=130
140 RS=16 : S=16 : C=1 : CS=1
150 '
160 SCREEN 0,0 : CLS 3
170 FOR R=RMIN TO RMAX STEP RS
180   FOR TH=0 TO 2*PI STEP 2*PI/S
190     X=320+R*COS(TH) : Y=100-R*P*SIN(TH)
200     CIRCLE (X,Y),8,C
210   NEXT TH
220   C=C+CS : IF C>7 THEN C=1
230 NEXT R
```

List 3.6a

```
140 RS=16 : S=16 : C=1 : CS=2
200     CIRCLE (X,Y),R/2,C
```

List 3.6b

```
140 RS=16 : S=16 : C=1 : CS=1
200     CIRCLE (X,Y),192-R,C
```

List 3.6c

```
140 RS=16 : S=16 : C=2 : CS=1
200     CIRCLE (X,Y),192-R,C
211   FOR TH=0 TO 2*PI STEP 2*PI/S
212     X=320+R*COS(TH) : Y=100-R*P*SIN(TH)
213     PAINT (X,Y),C-1,C
214   NEXT TH
```

List 3.6d

```
140 RS=14 : S=5  : C=2 : CS=1
200     CIRCLE (X,Y),192-R,C
211   FOR TH=0 TO 2*PI STEP 2*PI/S
212     X=320+R*COS(TH) : Y=100-R*P*SIN(TH)
213     PAINT (X,Y),C-1,C
214   NEXT TH
```

自分だけの曲線を描こう

sin 関数や cos 関数はグラフィックには欠かせない関数です．ここでは，数学的な関数の話はいっさいぬきにして，周期的に変化するという三角関数の性質を使って，いろいろな曲線を作ってみましょう．

まず，*List 3.7* を見てください．

190〜260 行の FOR〜NEXT ループは，200，210 行の sin，cos 関数の引数を，0 から 2π まで，0.01 ずつ変化させています．この π は，円周率を表し，3.14159……という値を持っています．sin 関数，cos 関数の，それぞれの引数を 0〜2π まで少しずつ変化させていくと，sin(A)，cos(A) は，*Figure 1* に示されるようなカーブを描きます．

sin(A)，cos(A) 関数の値は，2π のところで，ちょうどはじめの 0 の値に戻ります．これが，三角関数の 1 周期なのです．

変数 A の値を 0 から 2π まで変化させると，0と 2π では，sin(A)，cos(A) が同じ値をとることから，グラフの描きはじめと描き終わりが一致するということに気づきましたか．

220 行と 230 行は，変数 X，Y の値を，画面にちょうど収まる程度の大きさにしている部分です．もしここで値が大きすぎて，

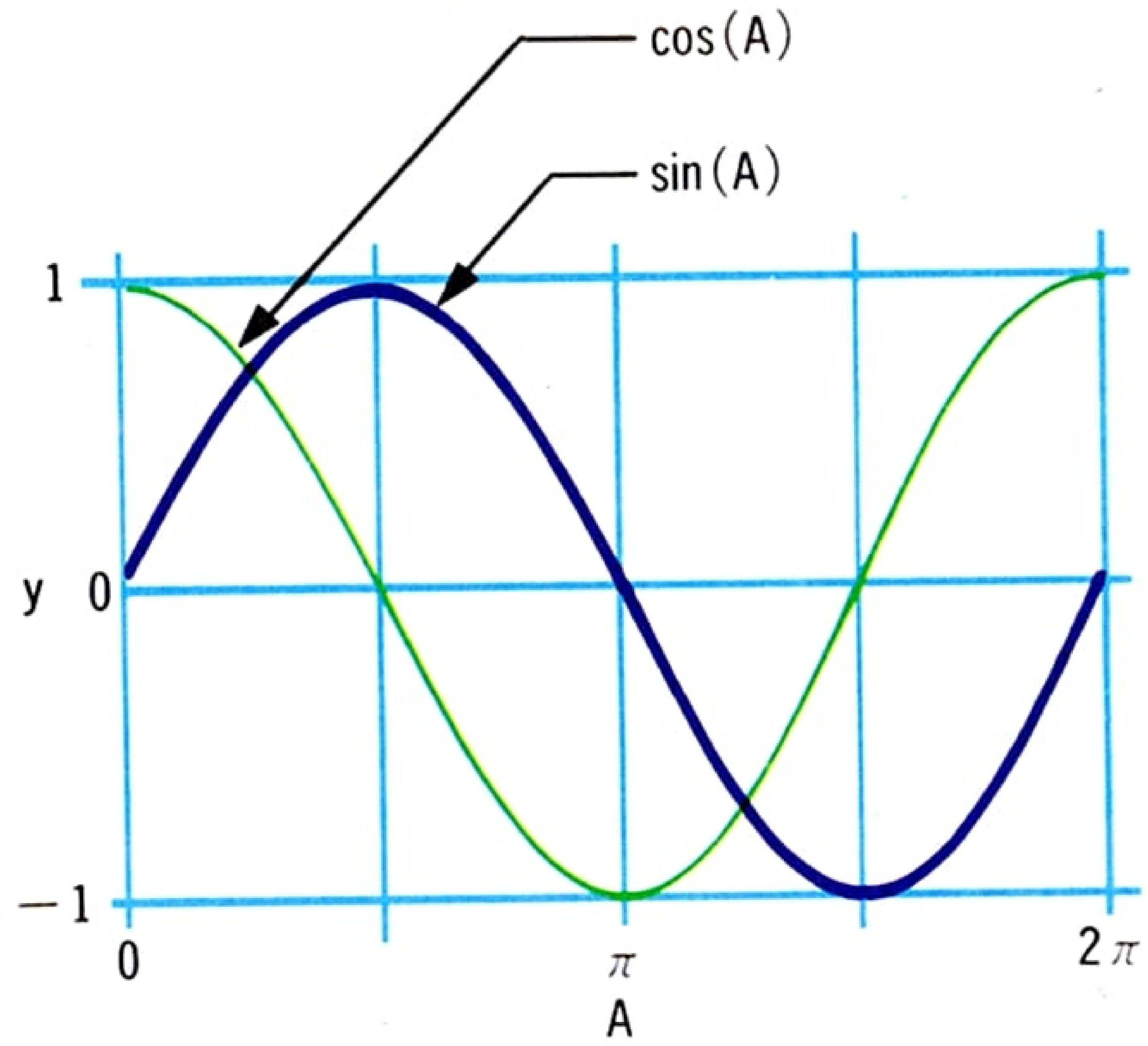
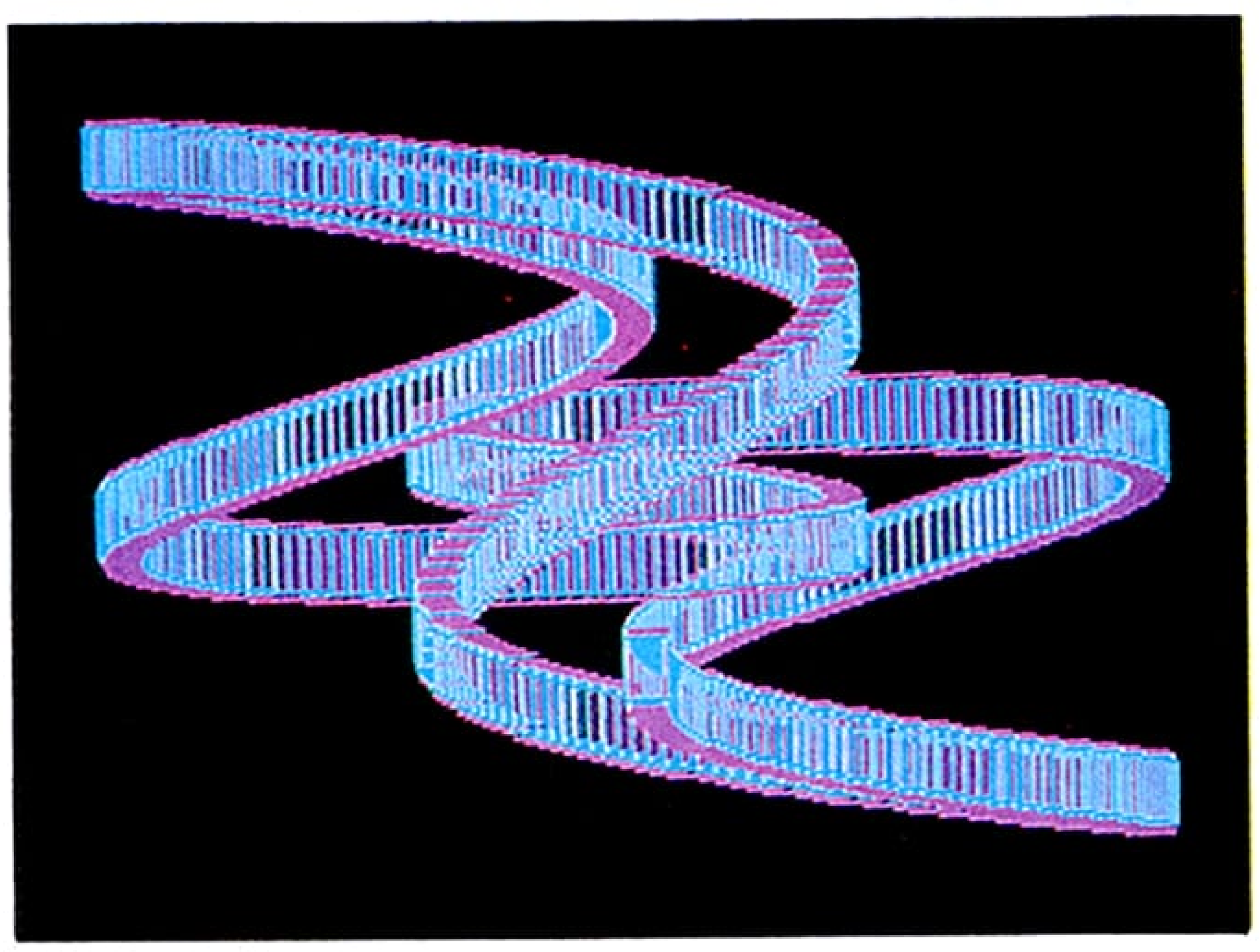
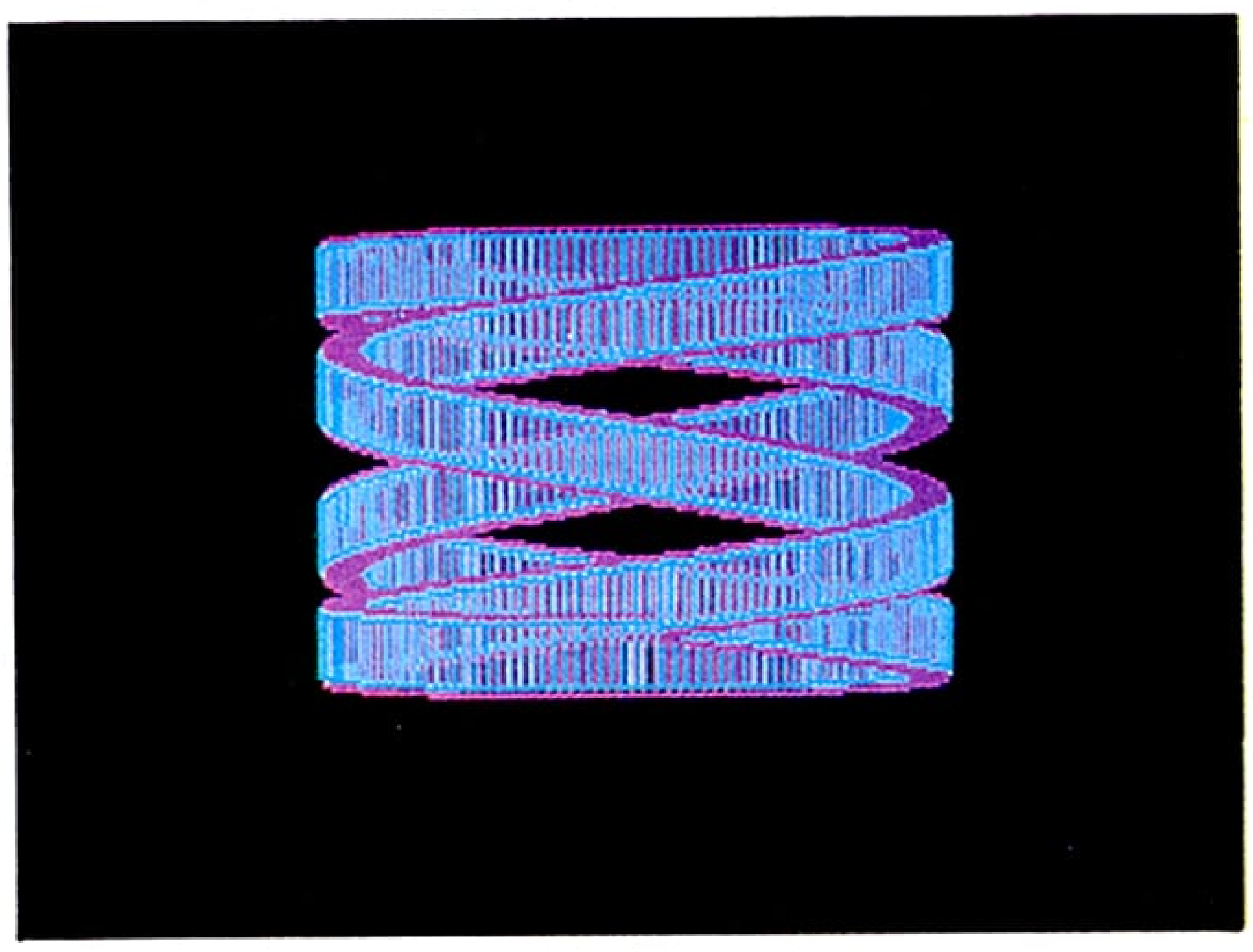


Figure 1





画面からはみ出してしまうような場合でも，PC-8801 mk IIでは，ワールド座標を使っているのでエラーは出ません．自分だけの曲線を作り出すといっても，何の手がかりもなしに行うことは難しいでしょう．そこで変数 X と Y をどのように書き変えたらよいのかを，*List 3.7a～e* に示しましたので試してみてください．200，210 行以外の行は，色や表示する形を変更するための行です．

List 3.7

```
100 ' List 3.7
110 '
120 PI=3.14159
130 XMIN=0   : YMIN=0
140 XMAX=639 : YMAX=199
150 XMID=(XMAX-XMIN)/2+XMIN  : YMID=(YMAX-YMIN)/2+YMIN
160 XW=(XMAX-XMIN)/4         : YW=(YMAX-YMIN)/4
170 '
180 SCREEN 0,0 : CLS 3
190 FOR TH=0 TO 2*PI STEP .01
200   X=COS(TH)-COS(TH*5)         } 表示する関数
210   Y=SIN(1.5+TH)-SIN(TH*2)     }
220   PX=X*XW+XMID
230   PY=Y*YW+YMID
240   LINE (PX-10,PY-10)-(PX+10,PY+10),3,B
250   LINE (PX-12,PY-8)-(PX+12,PY+8),5,B
260 NEXT TH
```

List 3.7a

```
200    X=SIN(TH*4)+COS(TH)
210    Y=SIN(TH)-COS(TH)
```

List 3.7b

```
200    X=(COS(TH*2)+COS(TH*5))*.8
210    Y=SIN(1.5+TH)-SIN(TH*2)
```

List 3.7c

```
200    X=SIN(TH*4)
210    Y=COS(TH)
```

List 3.7d

```
190 FOR TH=0 TO 2*PI STEP .02
200    X=SIN(TH)
210    Y=COS(TH)
```

List 3.7e

```
200    X=SIN(TH*4)
210    Y=COS(TH*3)
250    LINE (PX-8,PY-5)-(PX+8,PY+5),5,B
```

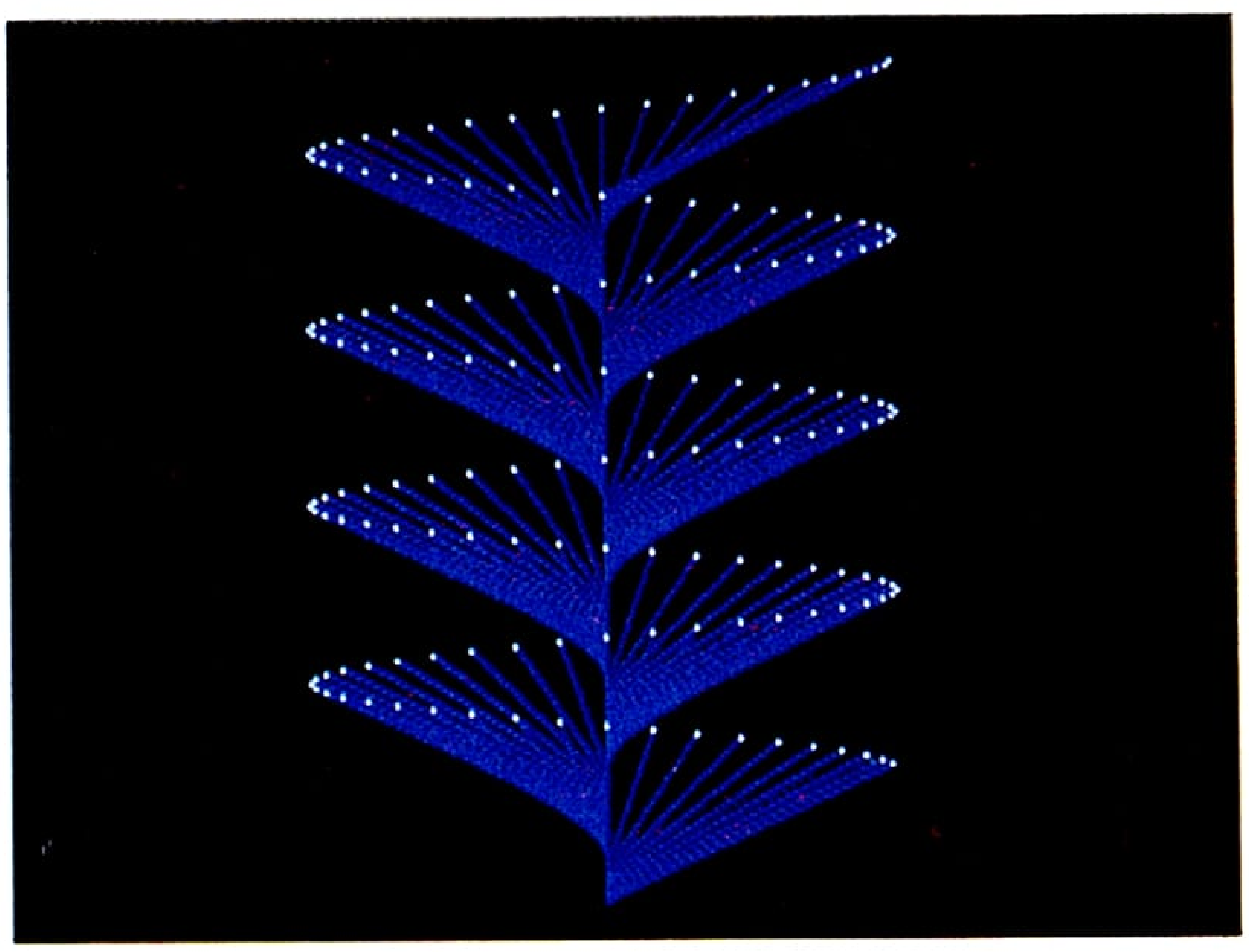
ヘリコイド曲線を描く

様々な曲線の中には，その描き方や形から，特定の名称がつけられているものがあります．もちろん，「円」もそうですが，サインカーブ，2次曲線，3次曲線，またちょっと難しいものでは，リサージュ図形，サイクロイドなんてのもあります．ここでいう，ヘリコイドも，名前のついた曲線の1つです．

ヘリコイド曲線は，「中心となる軸のまわりを，1点が少しずつ回りながら下へ向かって降りていく様子を，横から眺めたもの」と考えてください（*Figure 1* 参照）．

1点がある軸を中心に回転しながら降りていく場合に，それを横から見ると，*Figure 1b* のようになります．これは単なるコサインのカーブですが実はこれがヘリコイドの秘密なのです．このカーブと中心軸とを一定の間隔をあけて結びつけてやると，1点が軸を回りながら降下していくように見えるのです．*List 3.8* では，回転しながら降りていく点のＹ座標 YH と，結びつける中心軸上のＹ座標 YL は，YL−YH=30 という関係になっています（*Figure 2* 参照）．

ここで，cos 関数を用いてカーブを描きます．前にも述べたとおり，cos 関数は周期が 2π なので引数を0から 2π （π=3.



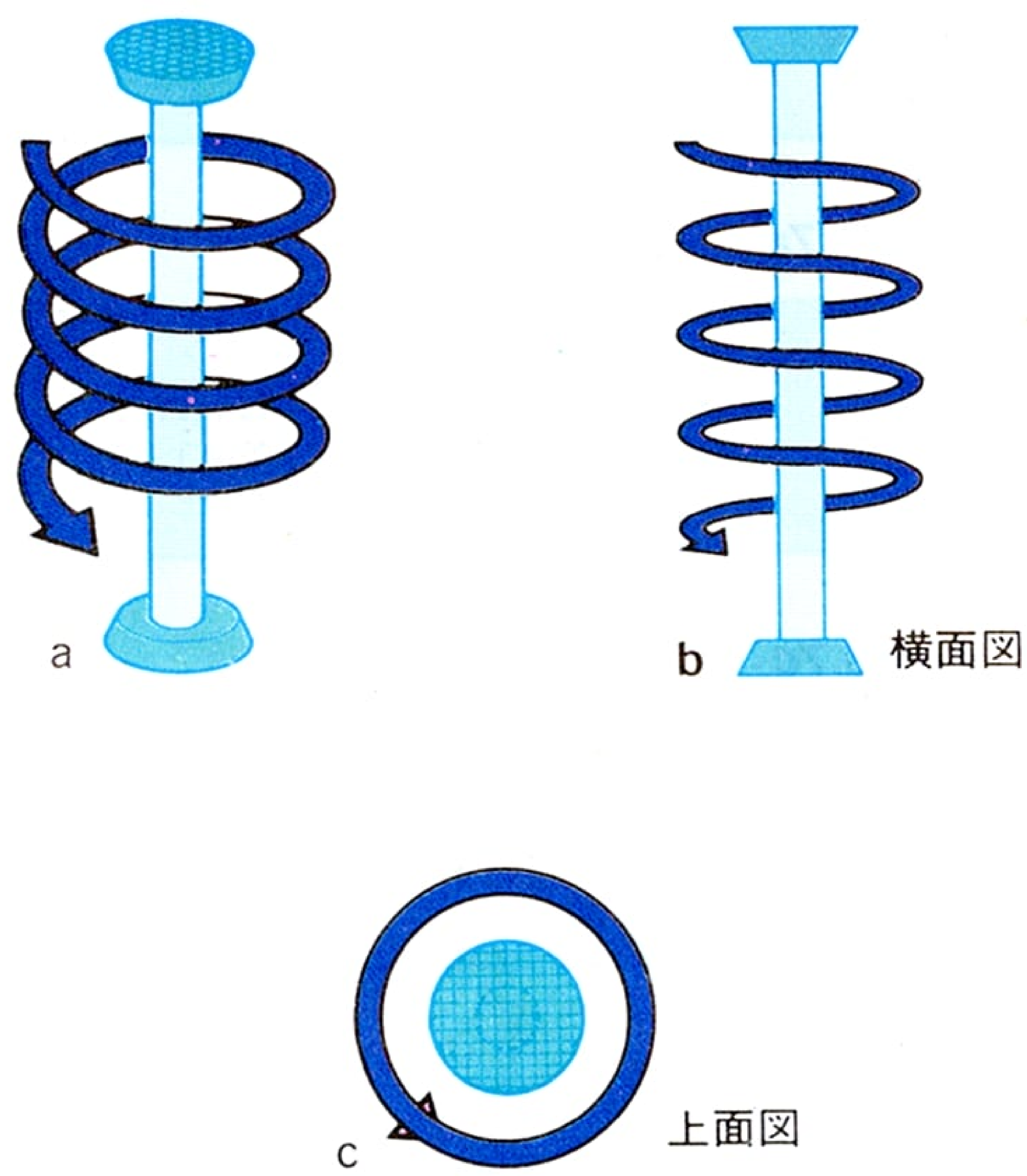


Figure 1

14159)まで少しずつ変化させると，ちょうど 0と 2π で同じ値になり，それを繰り返してカーブを描いていきます．

ここでは，210 行で TH を 0 から 8π まで変化させているので，計 4 周することになります．ここで，TH は π／20 ずつ動かしているので，1 周（2 π）するまでに，中心の軸と 40 個の点が結ばれることになります．

このプログラムでは，中心軸と点との距離を変数 R として，それを 180 ～ 280 行の FOR ～ NEXT で， RS おきに変化させています．従って，画面が進むにつれて，図形の横幅は広くなっていきます．

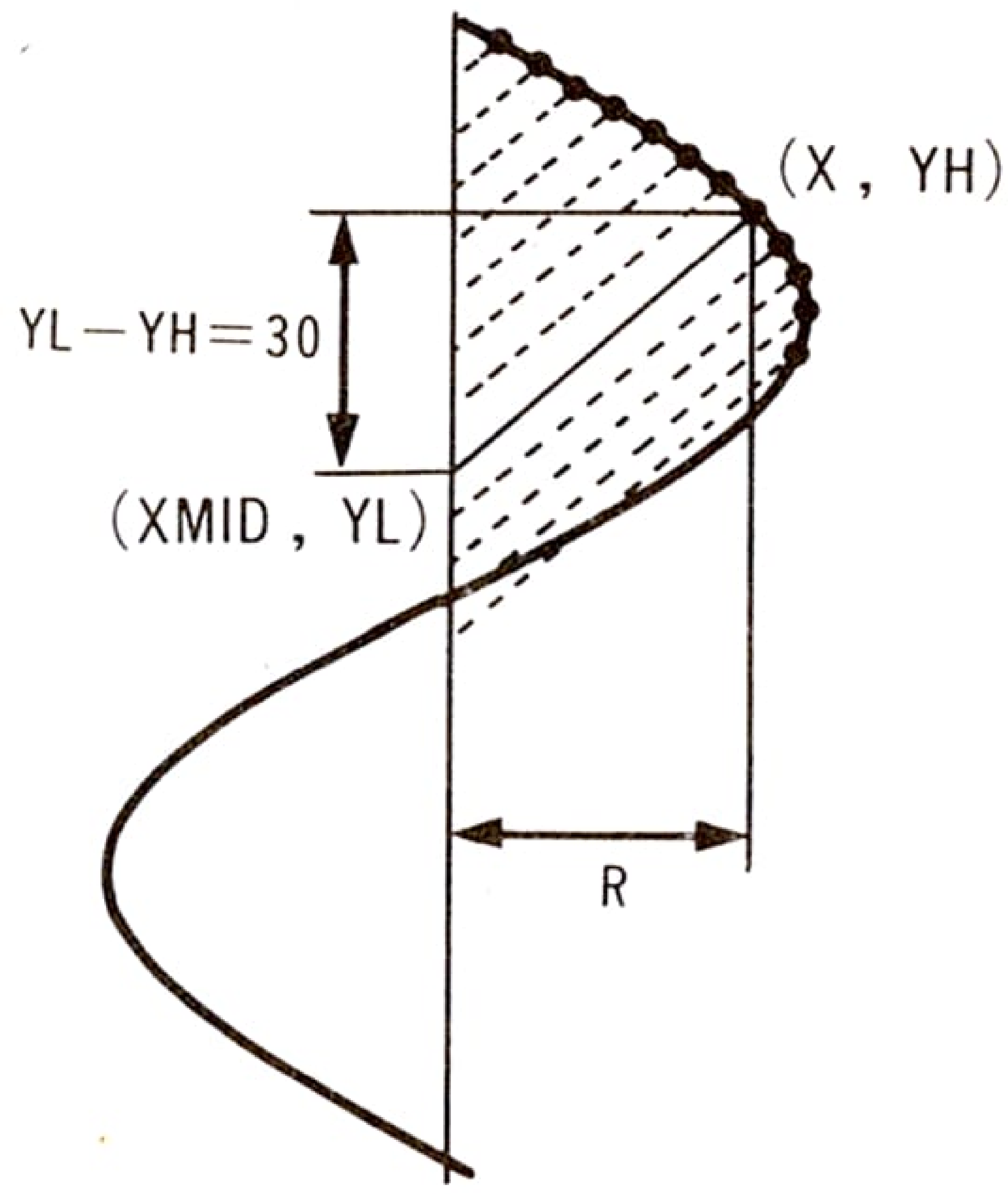


Figure 2

List 3.8

```
100 ' List 3.8
110 '
120 PI=3.14159
130 RMIN=20   : RMAX=260
140 XMID=320  : RS=60
150 C=1
160 '
170 SCREEN 0,0 : CLS 3
180 FOR R=RMIN TO RMAX STEP RS
190   LINE (XMID-R+RS,5)-(XMID+R-RS,195),0,BF
200   YL=35 : YH=5
210   FOR TH=0 TO 8*PI STEP PI/20
220     YL=YL+1 : YH=YH+1
230     X=R*COS(TH)+XMID
240     LINE (XMID,YL)-(X,YH),C
250     PSET (X,YH),7
260   NEXT TH
270   FOR I=1 TO 1000 : NEXT I
280 NEXT R
```

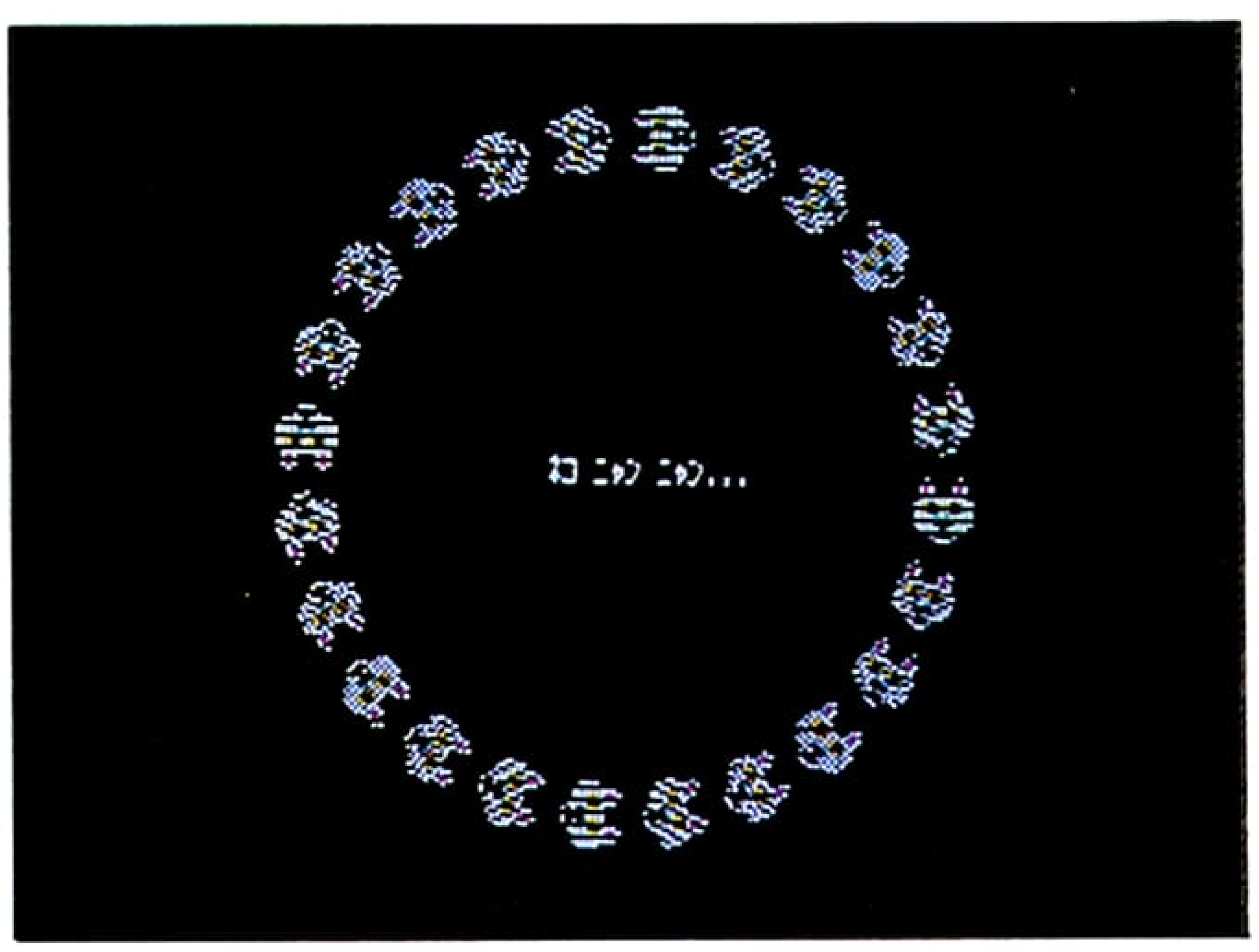
（210～270行：ヘリコイド曲線を描く）

ネコでも回そうか?

グラフィックの操作の中で，意外にめんどうなのは図形を傾けたり回転させたりすることです．例えばネコの顔を描くデータがここにあるとします．これを使って少し傾いたネコを表示しようと思ったら，まず顔の右端をちょっと下げて左端は上げ，頭のテッペンは右へずらしてアゴは左へ，というような操作が必要になるわけです．でも，こんなことをいちいち自分で計算してたらたいへんです．そこで，ここでは図形を回転させたときの位置を得るのに便利な式を紹介しましょう．

図形は，基本的には点の集まりです．今，回転の中心から(X，Y)だけ離れた位置にある点を1つ考えたとき，それが角度Tだけ回転した後の位置（XT，YT）は，

$$\begin{cases} XT=COS(T)*X-SIN(T)*Y \\ YT=SIN(T)*X+COS(T)*Y \end{cases}$$



という式で算出できます（*Figure 1* 参照）．この式を利用したのがこのプログラムです．

回転させるデータは121個の点から構成されます．これらは回転の中心位置から右へRだけ離れた場所に，x軸方向XS，y軸方向YSの距離を互いに保って並んでいます．この格子の左からI番目，上からJ番

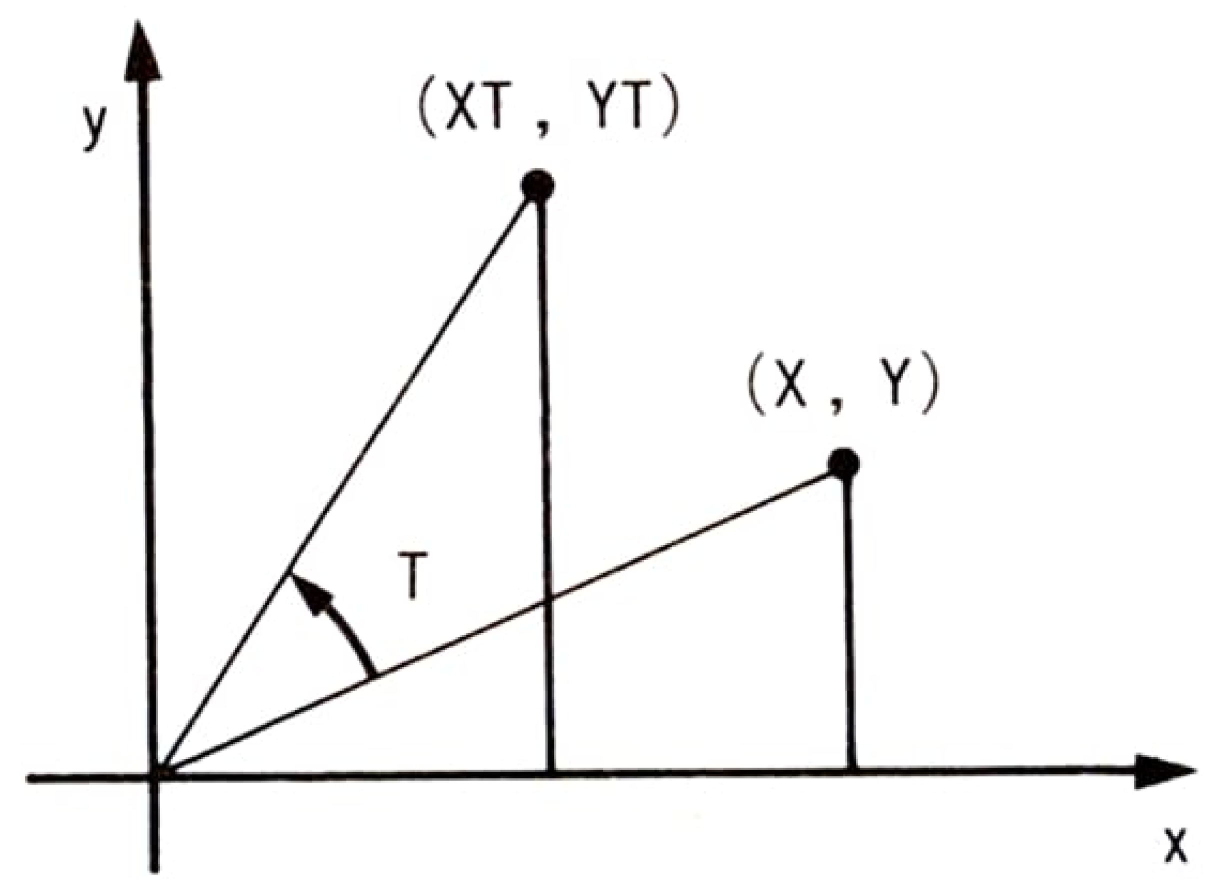


Figure 1

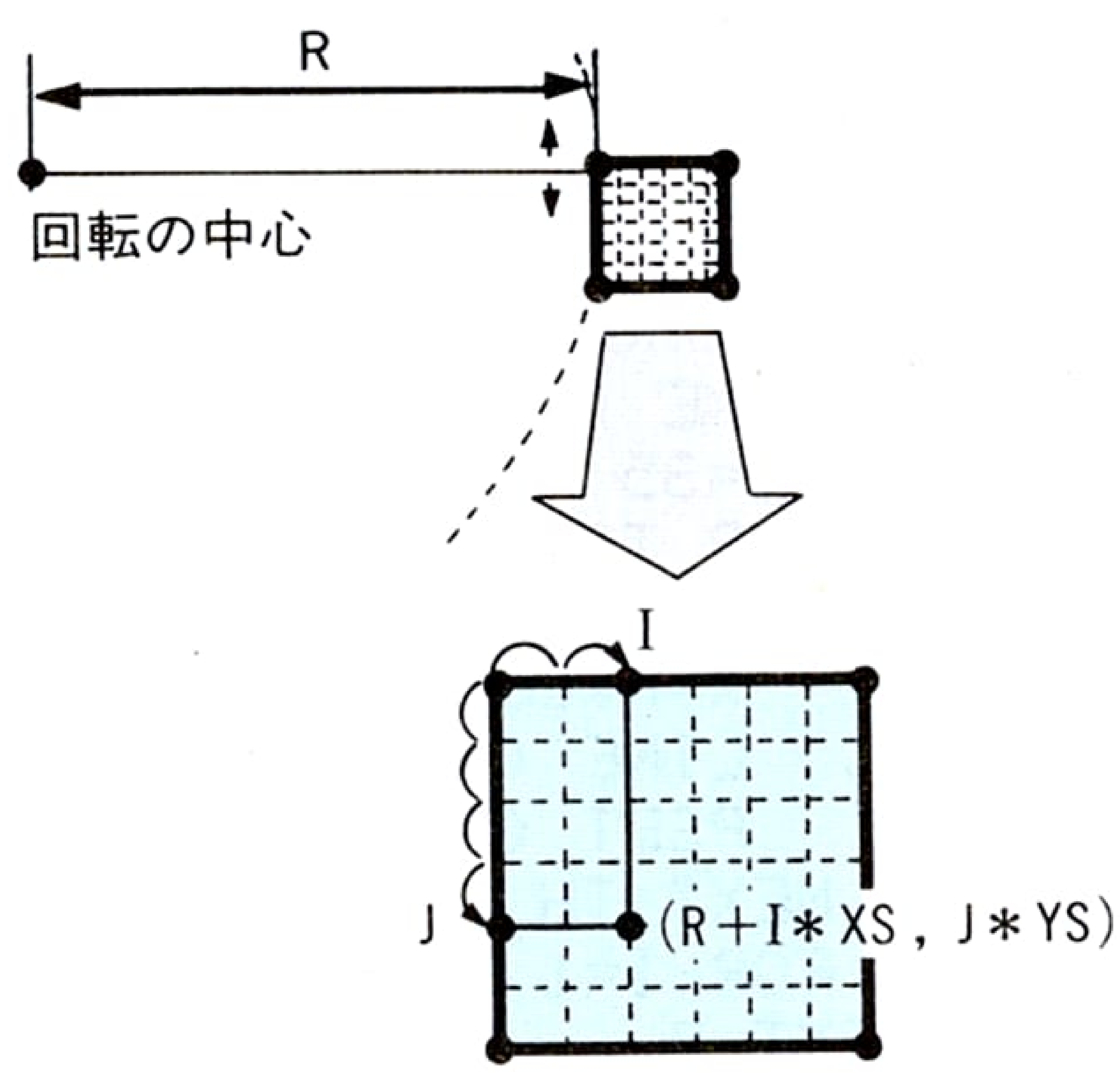


Figure 2

目に位置するドットの座標は，*Figure 2* のように（R+I＊XS，J＊YS）と表されます．さらにこの位置のドットの色は，配列 PC（I，J）に格納しておきます．

プログラムの 250～340行のループでは，DEGを変えて，各ドットの位置を計算して回転させ，PC(I，J)の色で点を打っています．

List 3.9

```
100 ' List 3.9
110 '
120 P=.5   : RPD=3.14159/180
130 R=150  : XS=3 : YS=3
140 D=10   : DIM PC(D,D)
150 '
160 FOR J=0 TO D
170   FOR I=0 TO D
180     READ PC(I,J)
190   NEXT I
200 NEXT J
210 '
220 WIDTH 80,25 : CONSOLE 0,25,0,1 : SCREEN 0,0 : CLS 3
230 LOCATE 35,12 : PRINT "ﾈｺ ﾆｬﾝ ﾆｬﾝ..."
240 '
250 FOR DEG=0 TO 359 STEP 15
260   RAD=DEG*RPD : CN=COS(RAD) : SN=SIN(RAD)
270   FOR J=0 TO D
280     Y=J*YS
290     FOR I=0 TO D
300       X=R+I*XS : XT=CN*X-SN*Y : YT=SN*X+CN*Y
310       PSET (320+XT,100+YT*P),PC(I,J)
320     NEXT I
330   NEXT J
340 NEXT DEG
350 END
360 '
370 DATA 0,0,7,0,0,0,0,0,7,0,0
380 DATA 0,7,3,7,0,0,0,7,3,7,0
390 DATA 0,7,3,7,0,0,0,7,3,7,0
400 DATA 0,7,7,7,7,7,7,7,7,7,0
410 DATA 7,7,6,0,6,7,6,0,6,7,7
420 DATA 7,7,6,0,6,7,6,0,6,7,7
430 DATA 7,7,7,7,5,7,5,7,7,7,7
440 DATA 7,7,7,7,0,0,0,7,7,7,7
450 DATA 0,7,7,7,7,0,7,7,7,7,0
460 DATA 0,0,0,7,0,0,0,7,0,0,0
470 DATA 0,0,0,0,7,7,7,0,0,0,0
```

データの読み込み（160～200行）

回転させるネコのデータ（370～470行）

3次元グラフィックス?

最近，TV コマーシャルやアニメ映画などでも，非常に美しい 3 次元グラフィックスが見られるようになりました．その美しさに感激して，パソコンで同様のことを行いたいと思っても，複雑な 3 次元→ 2 次元変換などが必要ですし，プログラムも膨大なものになってしまい，簡単には手が出せません．しかし，それほど複雑な理論を知らなくても「3 次元グラフィックス」は可能です．ここではそのテクニックを紹介します．

List 3.10 のプログラムでは，280，290 行で，3 次元データを 2 次元データに変換する処理を行っています．

ここで使っている式はどのようなものであるのか? まず *Figure 2* のような辺の長さが 1 の立方体を考えてみてください．この立方体の頂点 P（1，1，1）を画面に表示するには，原点からどれだけずらせばよいのでしょうか．

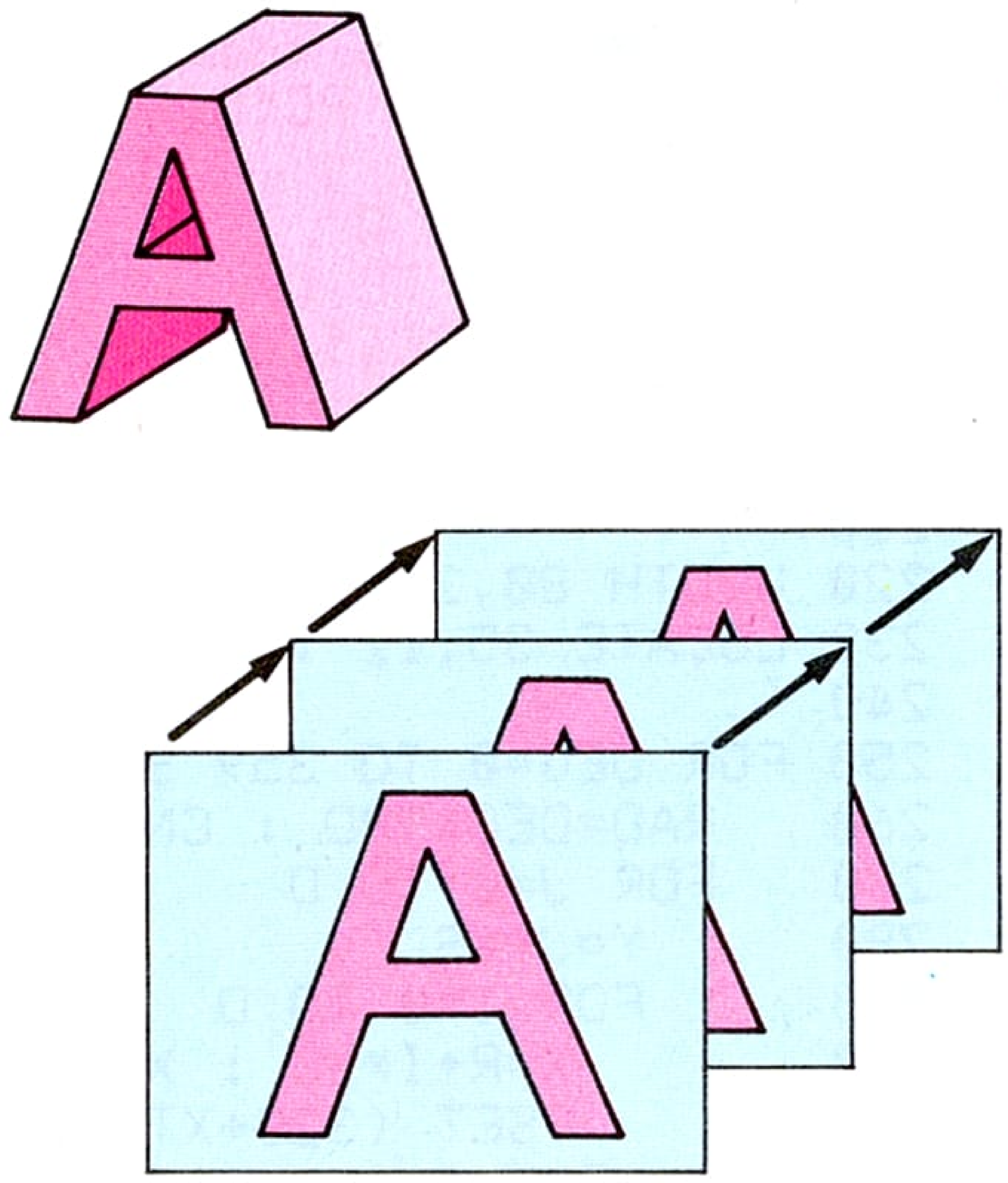
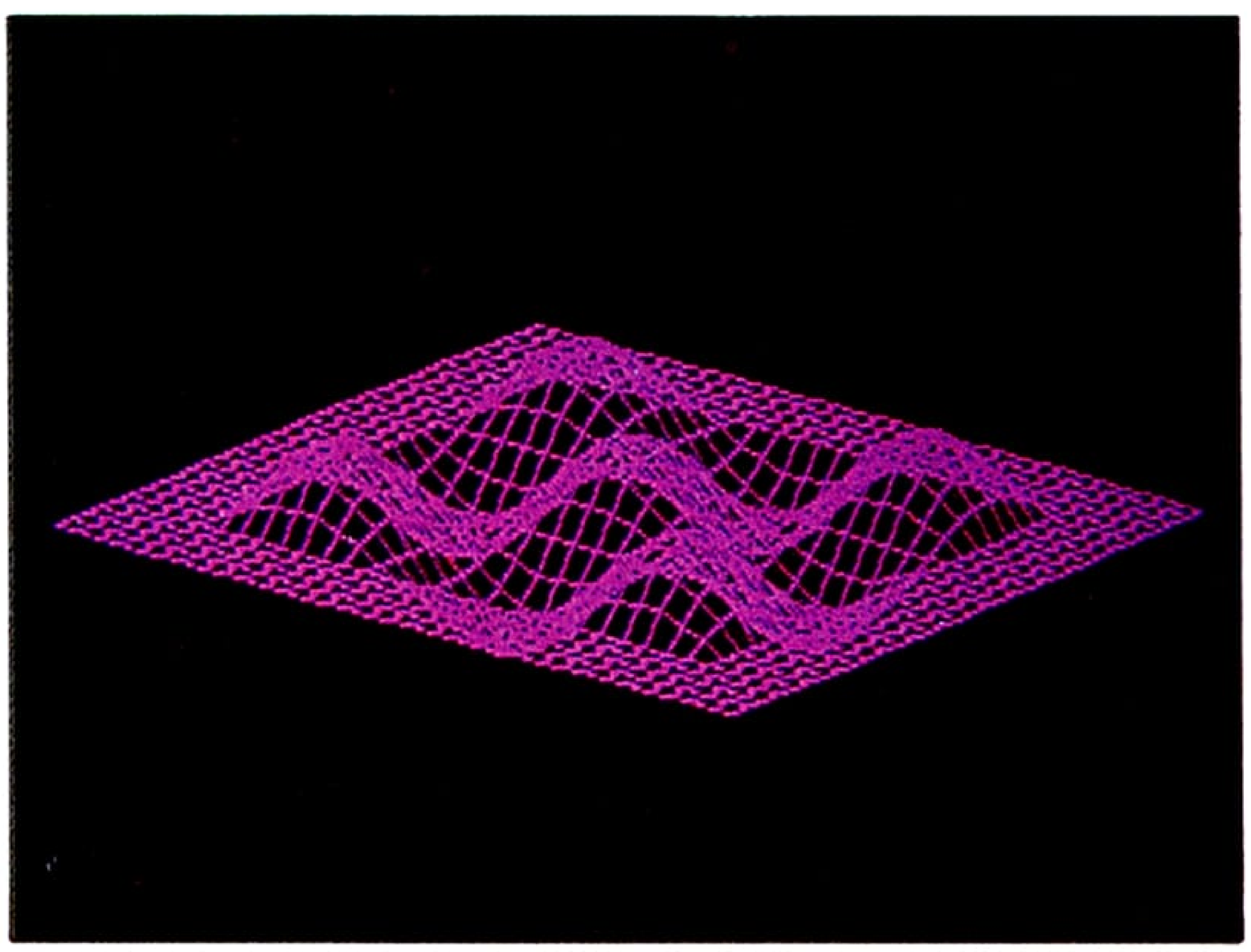
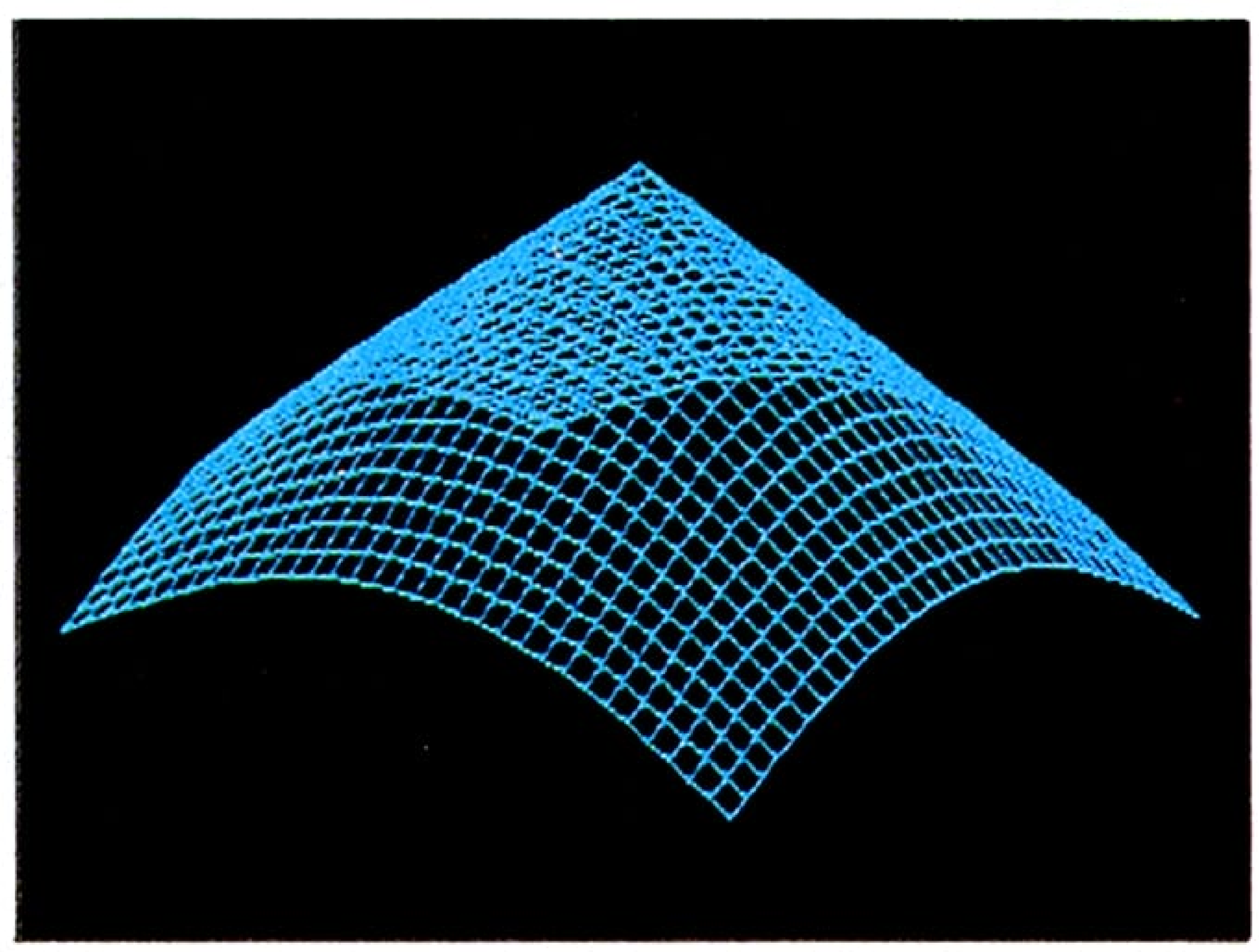


Figure 1





それには，いったん頭を３次元から２次元に切り換えて，*Figure 3*を見てください．

まず立方体を平面上に描き，そして図中に示したＫ１〜Ｋ５の長さを測ります．点Ｐの平面上での位置は，原点の（ＧＸ０，ＧＹ０)から右へＫ１，上へＫ２，左へＫ３，上へＫ４＋Ｋ５，だけ離れているわけですから，(ＧＸ０＋Ｋ１−Ｋ３，ＧＹ０−Ｋ２−Ｋ４−Ｋ５)と表すことができます．

そしてまた，このＫ１〜Ｋ５という値によって，もとの立方体で（ｘ，ｙ，ｚ）に位置していた点を画面に表示する場合の座標も，*Figure 4* に示すように計算できます．プログラムの280，290行で使っている計算式は，このようにして導かれたものです．

さて，プログラムを実行させるとわかるように，立体の各点は網目でつながっています．この網目は *Figure 5* に示すように，1つ前のカーブ上のそれぞれの点の位置を，配列Ｘ(Ｉ)，Ｙ(Ｉ) に記憶しておき，次の位置の点を結んでいくときに，この配列変数の値を参照して，つないで作られているのです．

このように３次元の物体を網目によって表す方法は，ワイヤ・フレーム法と呼ばれます．

ここで 88 ページの写真を見てください．見た目には３次元の物体のように見えます

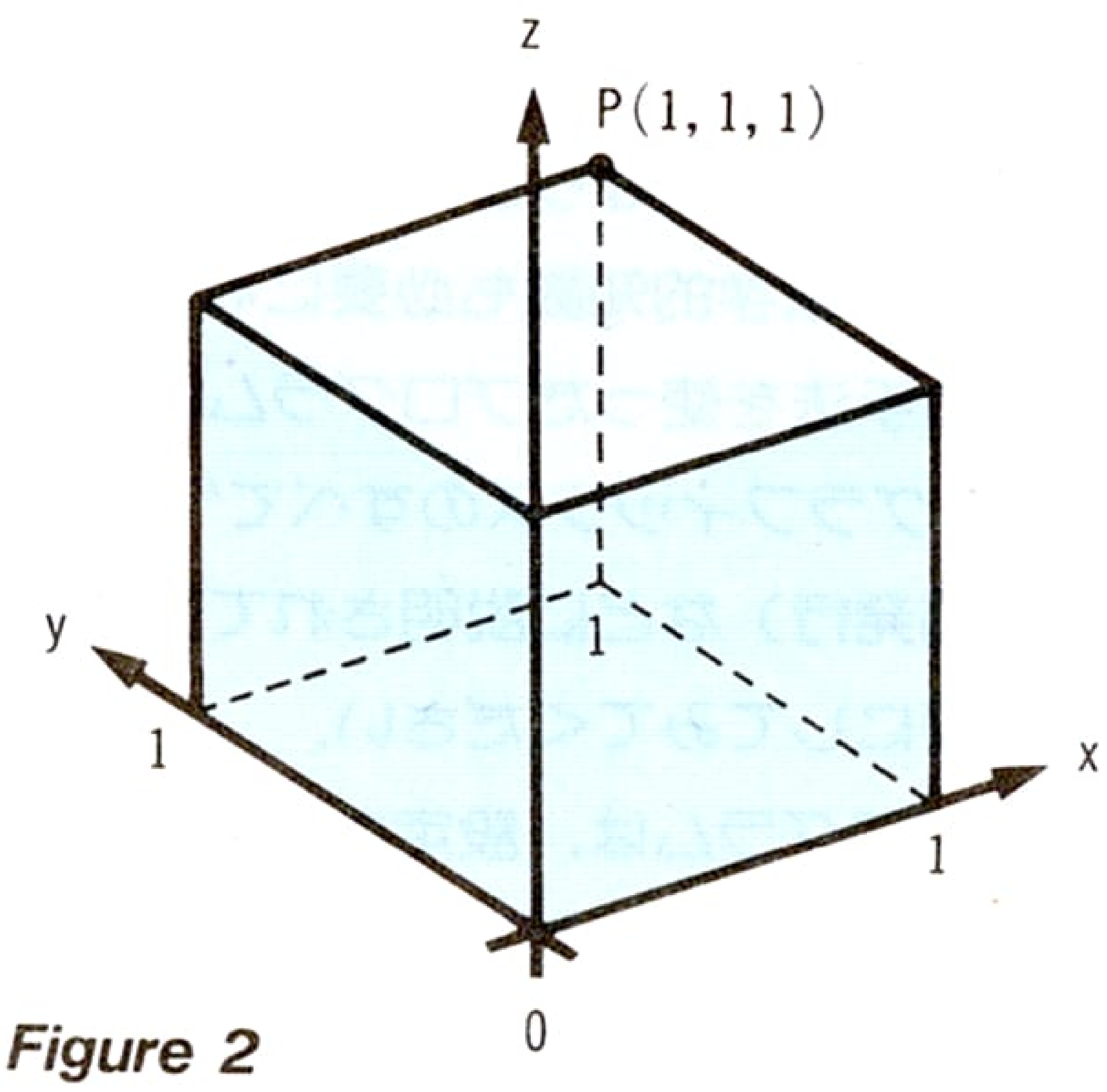


Figure 2

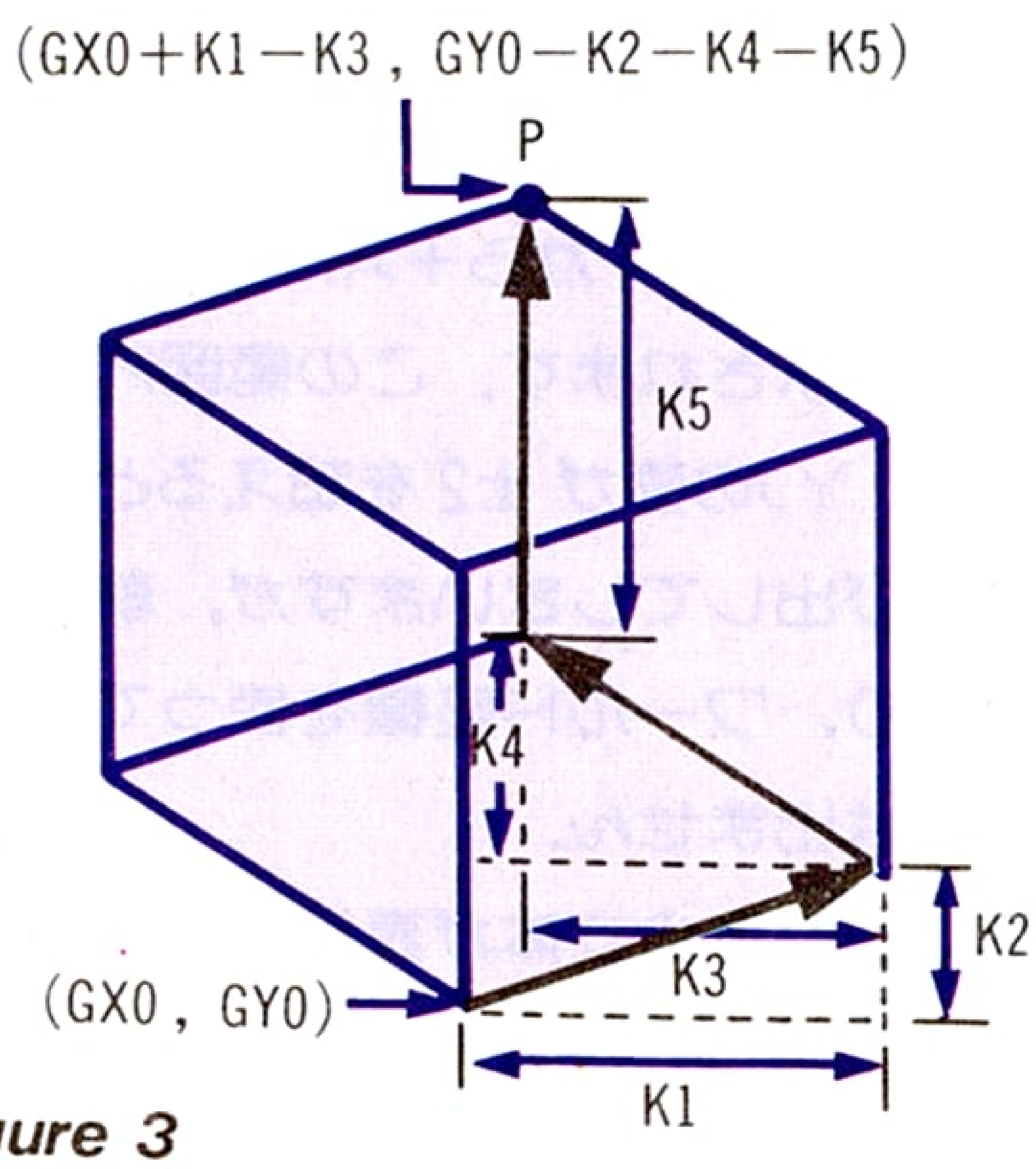


Figure 3

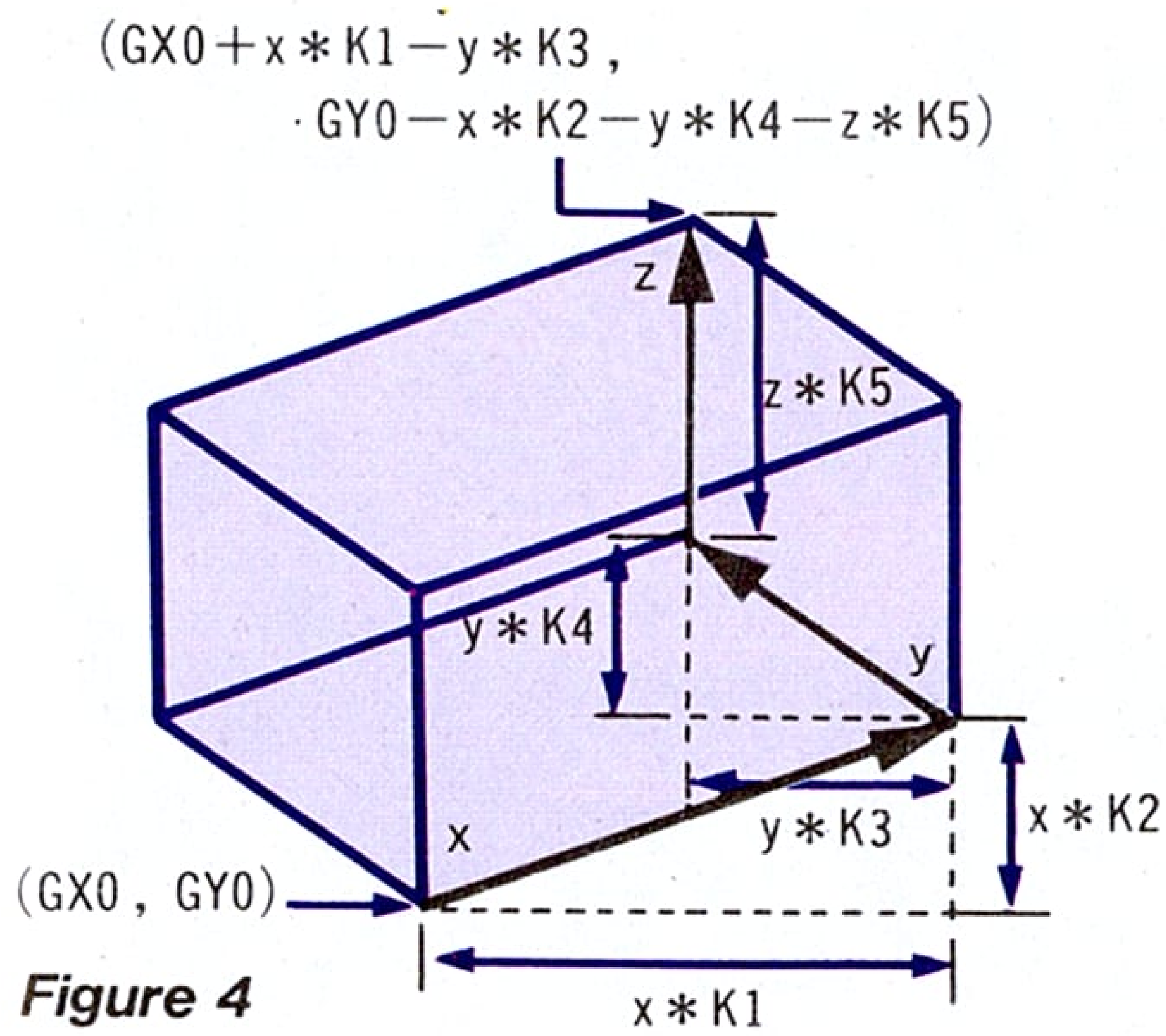


Figure 4

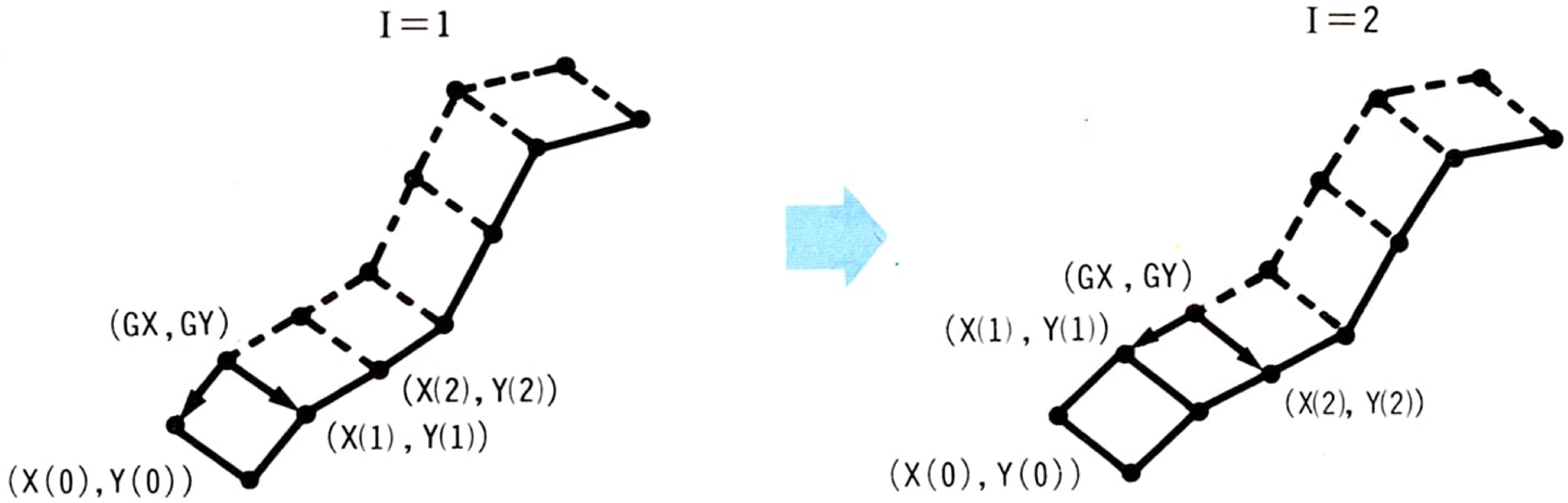
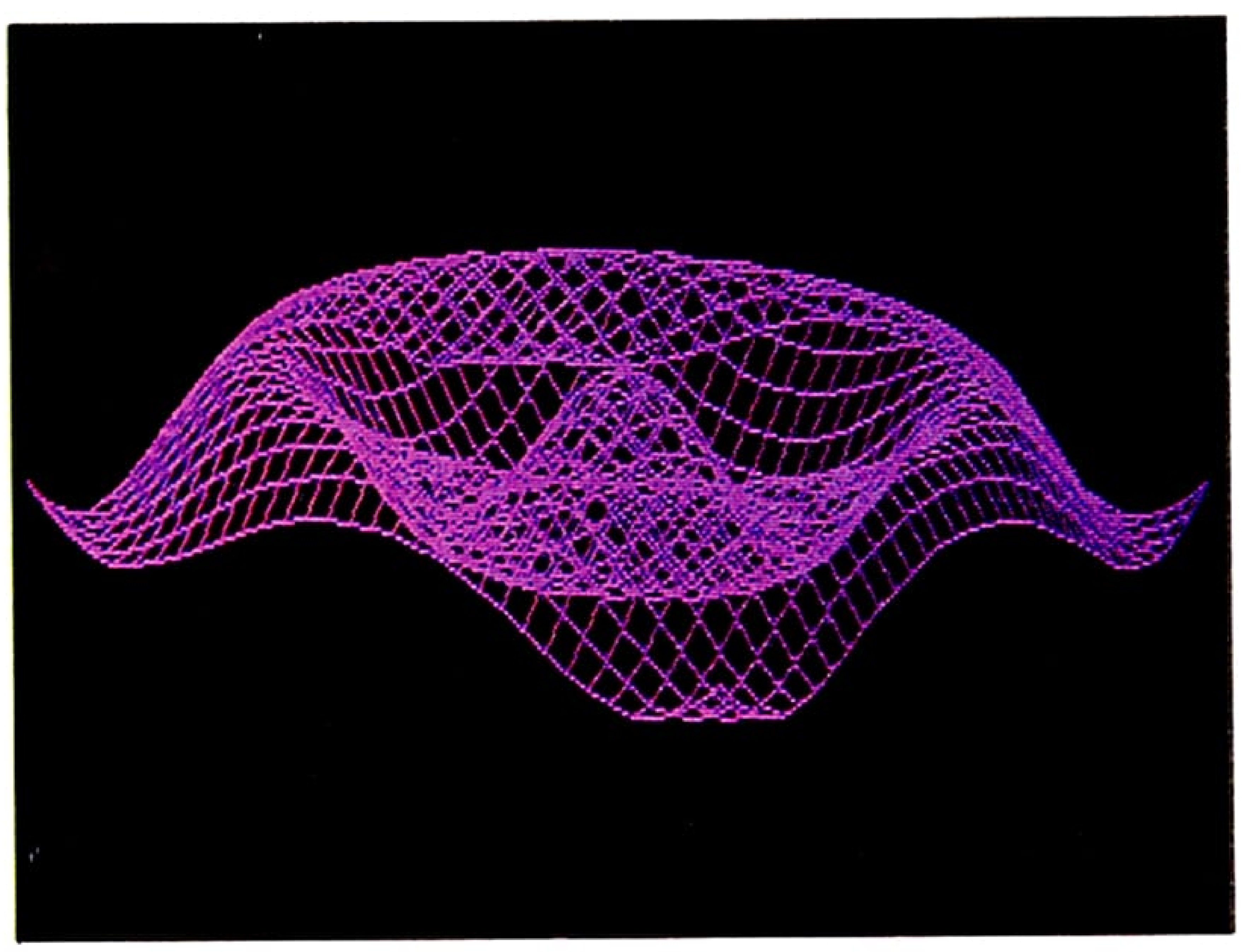


Figure 5　GX，GYが新しい X(1), Y(1)の値となって次へ進む

が，どこか奥行きがないように感じられませんか? ふつう手前にあるものほど大きく見えなければならないのですが，このプログラムでは，手前に見える山（こぶ）も奥に見える山（こぶ）も同じ大きさで表しているため奥行きがそれほど感じられないのです．この奥行きも考えに入れた本格的な三次元グラフィックを行うには，まず物体を見おろす方向と角度を決め，それに従って物体の座標データを回転し，透視変換し，クリッピングし……というような複雑な手順を踏まなくてはなりません．もちろん，高度な?　数学的知識も必要になってきます．その手法を使ったプログラムは，("PC-8801 グラフィックスのすべて"，アスキー出版局発行）などに説明されていますから，参考にしてみてください．



このプログラムは，設定を変えれば他の立体を表示することもできます．最後にその方法を紹介しておきましょう．

表示される図形は 150 行の関数定義命令で定義しています．そしてその関数の表示の範囲を決めているのが 130 行の AREA という変数です．x 座標，y 座標とも，0 を中心として－AREA から＋AREA までが計算され，表示されます．この範囲内で関数 FNZ(X，Y)の値が ±2 を超えると画面外に線が飛び出してしまいますが，前にもいったとおり，ワールド座標を使っているのでエラーは出ません．

おもしろい形の立体が表示される設定の例を *List 3.10a, b* に示しました．お試しください．

List 3.10

```
100 ' List 3.10
110 '
120 IMAX=32 : JMAX=32
130 AREA=12
140 PI=3.14159 : PD=PI/6
150 DEF FNZ(X,Y)=COS(X*PD)*COS(Y*PD)*(ABS(X)<9)*(ABS(Y)<9)
160 '
170 DIM X(IMAX),Y(IMAX)
180 XS=2*AREA/IMAX : YS=2*AREA/JMAX
190 K1=125/AREA : K2=25/AREA : K3=176/AREA : K4=22/AREA : K5=20
200 C=3
210 '
220 SCREEN 0,0 : CLS 3
230 FOR J=0 TO JMAX
240   Y=(J-JMAX/2)*YS
250   FOR I=0 TO IMAX
260     X=(I-IMAX/2)*XS
270     Z=FNZ(X,Y)
280     GX=320+X*K1-Y*K3
290     GY=100-X*K2-Y*K4-Z*K5
300     IF I>0 THEN LINE (GX,GY)-(X(I-1),Y(I-1)),C
310     IF J>0 THEN LINE (GX,GY)-(X(I),Y(I)),C
320     X(I)=GX : Y(I)=GY
330   NEXT I
340 NEXT J
```

（150行）表示するグラフィックを表す関数

（280～290行）3次元データを2次元データに変換

List 3.10a

```
140 PI=3.14159 : PD=PI/5
150 DEF FNZ(X,Y)=COS(SQR(X*X+Y*Y)*PD)
200 C=3
```

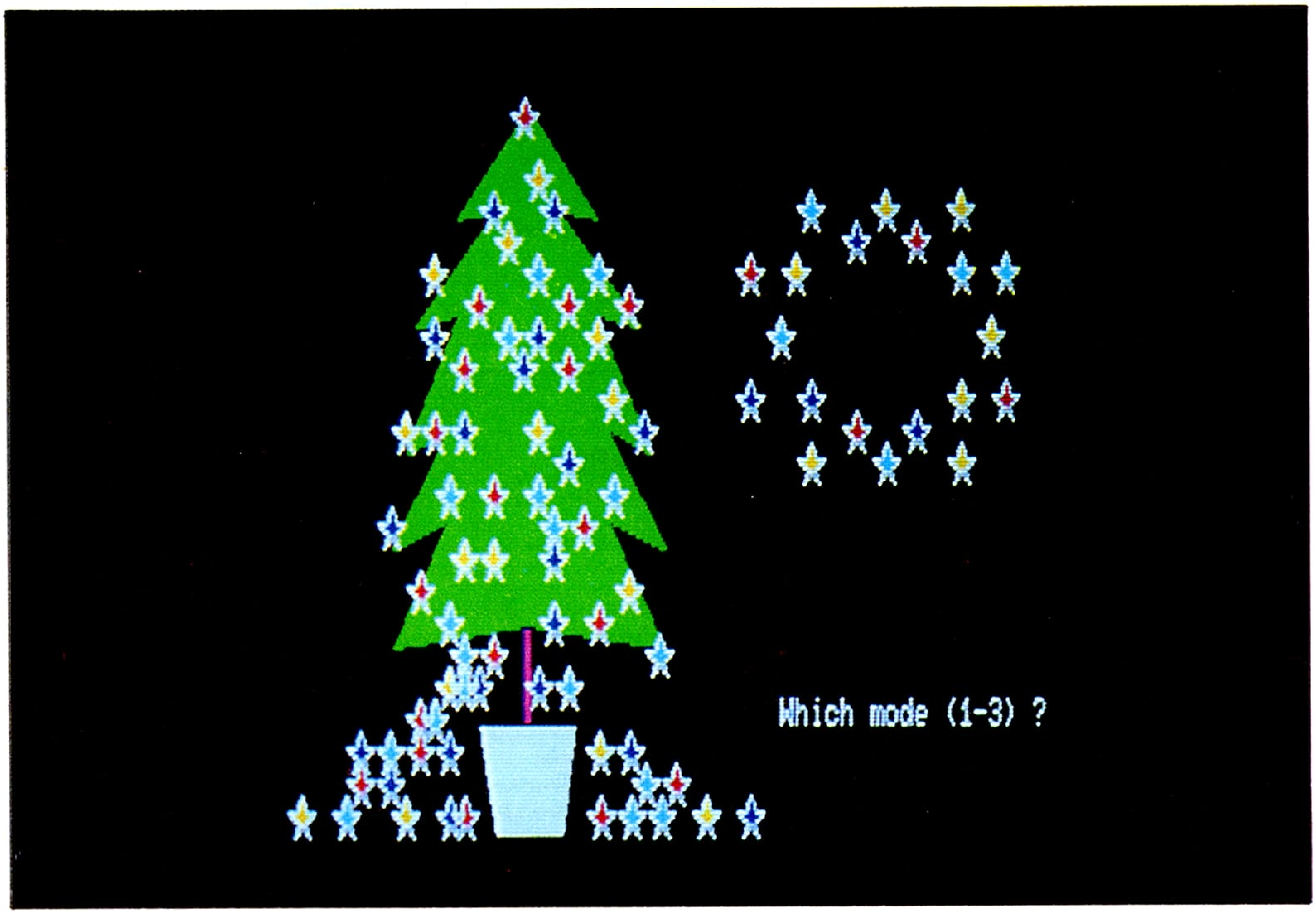
List 3.10b

```
130 AREA=4
140 '
150 DEF FNZ(X,Y)=4-SQR(X*X+Y*Y)
200 C=5
```

クリスマス・ツリー

PC-8801 mk IIには，グラフィック機能の１つとして，“パレット”というものが取り入れられています．注意していただきたいのは，パレットコードと色の対応を変えることによって，今までにそのパレットコードで出力された画面上のすべてのグラフィックの色が変化するということです．この機能を利用して，クリスマス・ツリーのデモンストレーション・プログラムを作ってみました．これは，星をツリーに飾りつけ，その星を点滅させるものです．

プログラムを実行すると，まずクリスマス・ツリーを描き，その右に十字形のカーソルを表示します．このカーソルは，カーソルキーによって任意の場所に移動できます．また，1～4のキーはそれぞれ青，赤，水色，黄色に対応していて，これらのキーを押すと，カーソルの位置に星が出力され



ます．これを繰り返してツリーを飾りつけ，画面が星でいっぱいになったら，リターンキーを押してください．画面に“Which mode(1−3)?”と出力されますから，1〜3のキーを押して，星の点滅の仕方を3つの中から選んでください．1〜3のモードがそれぞれどう点滅するかは，実際に実行してみてのお楽しみです．なお，ESC キーを押すことによって，再び点滅モードの選択に戻ることができます．プログラムを終了したいときはもう1度 ESC キーを押してください．

List 3.11

```
1000 ' List 3.11
1010 '
1020 WIDTH 80,25 : CONSOLE 0,25,0,1 : SCREEN 0,0 : CLS 3
1030 '
1040 RANDOMIZE VAL(RIGHT$(TIME$,2))
1050 STAR(1)=1 : STAR(2)=2 : STAR(3)=5 : STAR(4)=6
1060
1070 GOSUB *TREE
1080 GOSUB *STAR
1090 '
1100 *MENU
1110 FOR I=1 TO 4 : C(I)=STAR(I) : NEXT I
1120 CLS : LOCATE 40,20 : PRINT "Which mode (1-3) ?";
1130 I$=INPUT$(1)
1140   IF I$=CHR$(27) THEN FOR I=1 TO 7:COLOR=(I,I):NEXT I:END
1150   IF I$<"1" OR I$>"3" GOTO 1130
1160   LOCATE 40,20 : PRINT SPACE$(30);
1170   ON VAL(I$) GOSUB *MODE1,*MODE2,*MODE3
1180 GOTO *MENU
1190 '
1200 *TREE
1210 '
1220 FOR I=1 TO 7 : COLOR=(I,I) : NEXT I
1230 CW=7 : GOSUB 1300
1240 PAINT (186,20),4,7
1250 PAINT (186,150),3,7
1260 PAINT (186,170),7,7
1270 CW=0 : GOSUB 1300
1280 RETURN
1290 '
```

初期設定

表示する星のパレットコードを設定

キー入力処理ルーチン

クリスマス・ツリーを塗るサブルーチン

```
1300 RESTORE *TREEDATA : READ XO,YO
1310 FOR I=1 TO 14
1320  READ XN,YN
1330  LINE (186+XO,YO)-(186+XN,YN),CW
1340  LINE (186-XO,YO)-(186-XN,YN),CW
1350  XO=XN : YO=YN
1360 NEXT I
1370 RETURN
1380 '
1390 *TREEDATA
1400 DATA  0, 10,  40, 40,  19, 38,  64, 68,  38, 64
1410 DATA 68,100,  44, 93,  76,124,  44,116,  74,148
1420 DATA -2,142,  -2,166,  26,166,  18,194,   0,194
1430 '
1440 *STAR
1450 XN=35 : YN=10
1460 XS(1)=1 : XS(2)=-1 : XS(3)=0  : XS(4)=0
1470 YS(1)=0 : YS(2)=0  : YS(3)=-1 : YS(4)=1
1480 KY$="1234"+CHR$(28)+CHR$(29)+CHR$(30)+CHR$(31)+CHR$(13)
1490 '
1500 *CURSOR
1510 LOCATE XO,YO: PRINT " "; : LOCATE XN,YN : PRINT "+";
1520 XO=XN : YO=YN
1530 '
1540 *KYINP
1550   I$=INKEY$ : IF I$="" GOTO 1550
1560   I=INSTR(KY$,I$)
1570   IF I>=1 AND I<=4 THEN GOSUB *DOT
1580   IF I>=5 AND I<=8 THEN DR=I-4 : GOTO *CHECK
1590   IF I=9 THEN RETURN
1600 GOTO *KYINP
1610 '
1620 *CHECK
1630 XN=XO+XS(DR) : IF XN<1 OR XN>78 GOTO *KYINP
1640 YN=YO+YS(DR) : IF YN<1 OR YN>23 GOTO *KYINP
1650 GOTO *CURSOR
```

クリスマス・ツリーを描くサブルーチン（1300〜1370）

クリスマス・ツリーのデータ（1400〜1420）

カーソルの表示（1500〜1520）

カーソルキーによるカーソル移動ルーチン（1540〜1650）

星を描く

```
1660
1670 *DOT
1680 X=XO*8 : Y=YO*8 : POINT (X,Y)
1690 LINE -STEP(-3,4),7 :LINE -STEP(-4,0),7: LINE -STEP(5,3),7
1700 LINE -STEP(-2,3),7 :LINE -STEP(4,-3),7: LINE -STEP(4,3),7
1710 LINE -STEP(-2,-3),7:LINE -STEP(5,-3),7:LINE -STEP(-4,0),7
1720 LINE -STEP(-3,-4),7:PAINT (X,Y+3),STAR(I),7
1730 RETURN
1740 '
1750 *MODE1
1760 FOR I=1 TO 4 : T(I)=0 : COLOR=(C(I),0) : NEXT I
1770 R=INT(RND*4)+1
1780  IF T(R)=0 THEN T(R)=INT(RND*30)+1 : BEEP 1 : BEEP 0
1790  FOR I=1 TO 4
1800    IF T(I)>0 THEN T(I)=T(I)-1 : COLOR=(C(I),C(I))
1810    IF T(I)=0 THEN COLOR=(C(I),0)
1820  NEXT I
1830 IF INKEY$<>CHR$(27) GOTO 1770 ELSE RETURN
1840 '
1850 *MODE2
1860 C(0)=C(1): C(1)=C(2): C(2)=C(3): C(3)=C(4): C(4)=C(0)
1870   FOR I=1 TO 4
1880     COLOR=(STAR(I),C(I))
1890   NEXT I
1900   FOR T=1 TO 100 : NEXT T
1910 IF INKEY$<>CHR$(27) GOTO 1860 ELSE RETURN
1920 '
1930 *MODE3
1940 FOR I=1 TO 4
1950   COLOR=(C(I),0)
1960   FOR T=1 TO 100 : NEXT
1970   COLOR=(C(I),C(I))
1980 NEXT I
1990 IF INKEY$<>CHR$(27) GOTO 1940 ELSE RETURN
```

※

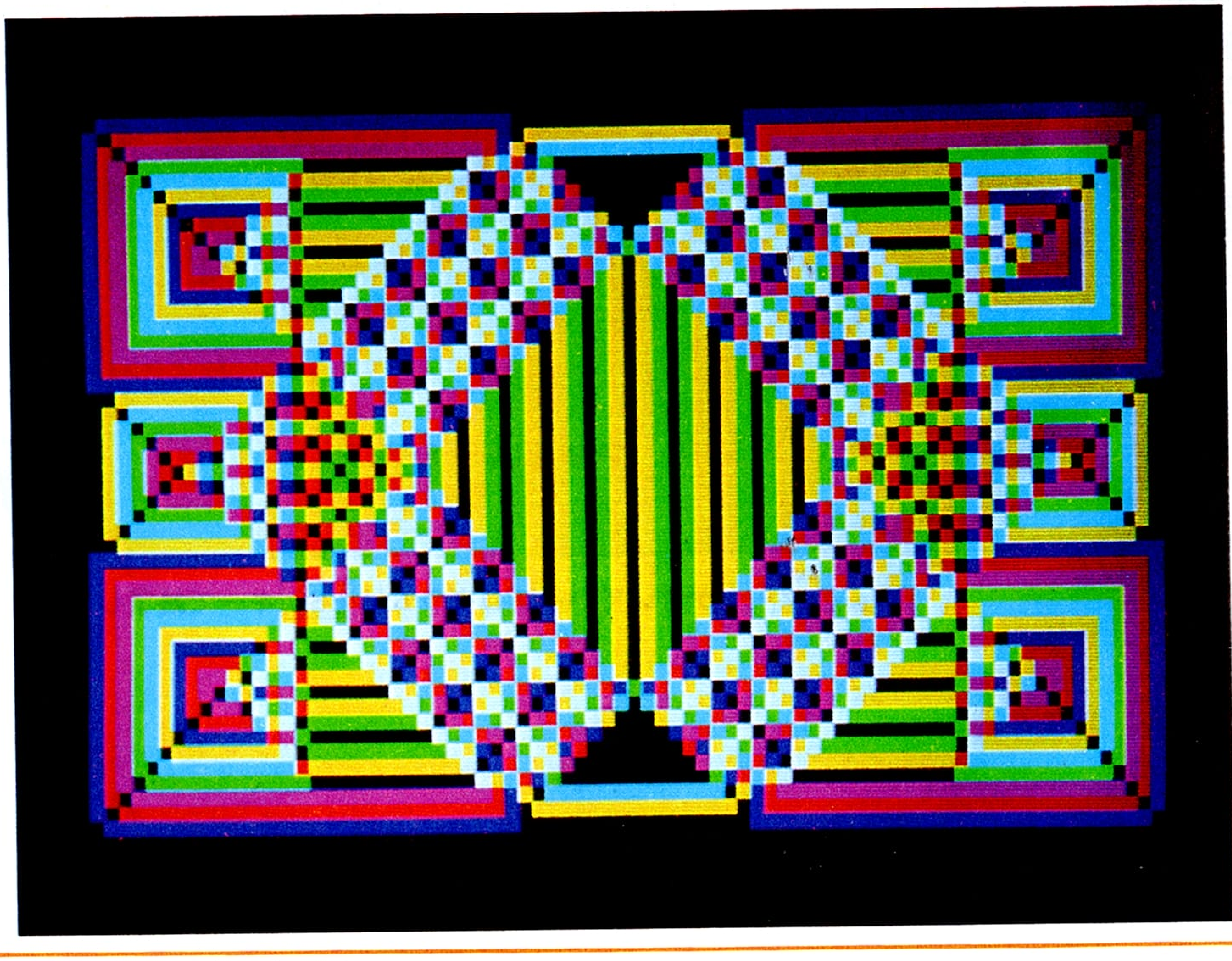
※ このマークが付いているプログラムは ESC キーで終了してください

XORは色の上位に色を作らず

昨今の若い読者の方にはお馴染みでないかもしれませんが，我が国には昔から「三すくみ」という名で知られる関係を保ち，互いに牽制しあう三種の生物集団が存在します．

まずは，ナメクジ．こいつが何に弱いったって，もう蛙が大の苦手．というのも，蛙はナメクジが大好物で，姿を見るやいなやパクリとやってしまうため．ナメクジだって食われたかぁありませんよね．一方，その蛙が蛇には勝てないってことは，「蛇に睨まれた蛙のような」という言葉もあることですし，皆さん知っておいででしょう．ところが，それじゃあ最強の存在かと思われた蛇が，実はナメクジが一番の天敵というのだから世の中は皮肉なものです（もっとも，この最後の話は少しマユツバですが）．

ある日，この三者が顔を合わせた途端，皆いっせいにすくみあがって一歩も動けなくなった．これが「三すくみ」の語源である，なんてのは〝講釈師見てきたような嘘を言い〞，のたぐいですかしらん．



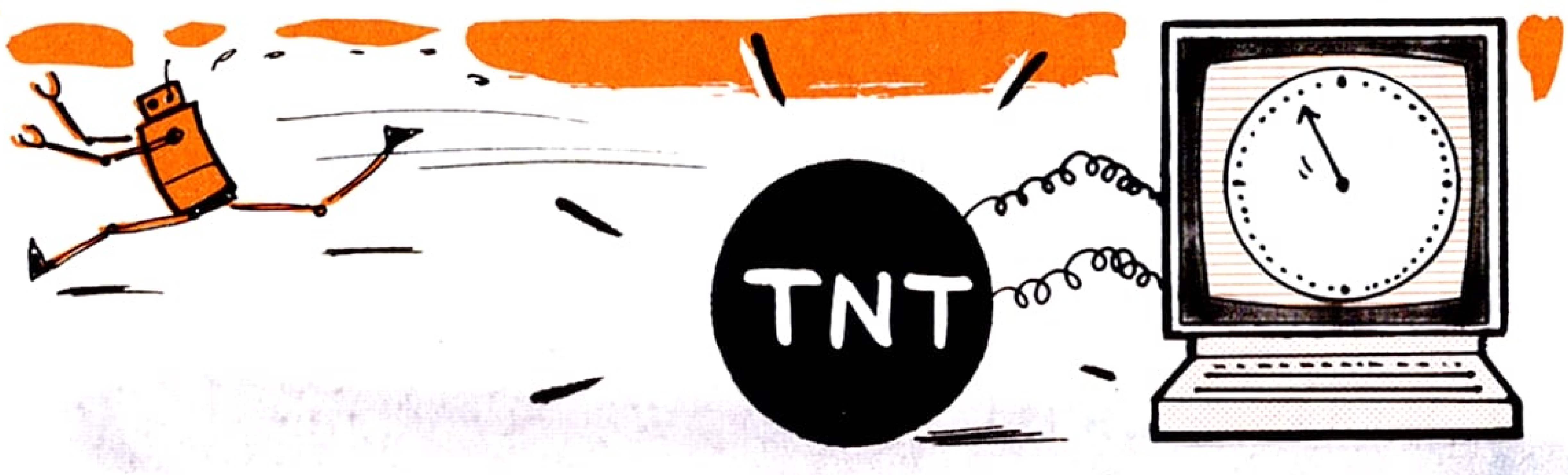
この「三すくみ」的な面白さの一端は，実はグラフィックの中にも見ることができます．PC-8801mkIIのグラフィック命令には，使用する色と下地に塗ってある色との間で，論理演算(AND, OR, XOR)を行う機能があります．このうち，AND機能を使う場合には，0から7までのパレットコードのなかでも0が最も強力であり，他のどの色とAND演算を行っても結果はやはり0になります．またOR演算では7が強く，例えば乱数で色を決めた四角形をOR機能を用いて次々と重ねていくと，画面は最終的には7の色（パレットが変えられてなければ，白ですね）でいっぱいになってしまいます．

けれどもXORという機能に関しては，ある色が最も強いなんていうことがない．ですから，どんな色を塗り重ねていっても，画面は一色に染まりません．あとからあとから異なった色が現れます．これですよ，これ．この性質を利用すると，たいへん美しいグラフィックパターンを描き出すことができます．その例が次のプログラムです．どうぞお試しください．

```
100 ' List Ex.3
110 '
120 CONSOLE 0,25,0,1 : WIDTH 80,25
130 SCREEN 0,0 : CLS 2
140 '
150 DIM D%(3)
160 SCREEN 1,0,0,1 : LINE (0,0)-(7,3),1,BF
170 GET (0,0)-(7,3),D% : SCREEN 0,0 : CLS 2
180 '
190 X=0 : Y=0 : I=8 : J=4 : K=1 : C=1
200 '
210 *LOOP
220 FOR XX=X TO 624-X STEP SGN(624-2*X)*ABS(I)
230   PUT (XX,Y),D%,XOR,C,0 : PUT (XX,192-Y),D%,XOR,C,0
240 NEXT XX
250 FOR YY=Y TO 192-Y STEP SGN(192-2*Y)*ABS(J)
260   PUT (X,YY),D%,XOR,C,0 : PUT (624-X,YY),D%,XOR,C,0
270 NEXT YY
280 IF X+I<0 OR X+I>624 THEN I=-I
290 IF Y+J<0 OR Y+J>192 THEN J=-J
300 X=X+I : Y=Y+J : C=C+K
310 IF C>6 THEN C=1 ELSE IF C<1 THEN C=6
320 GOTO *LOOP
```

【4章】

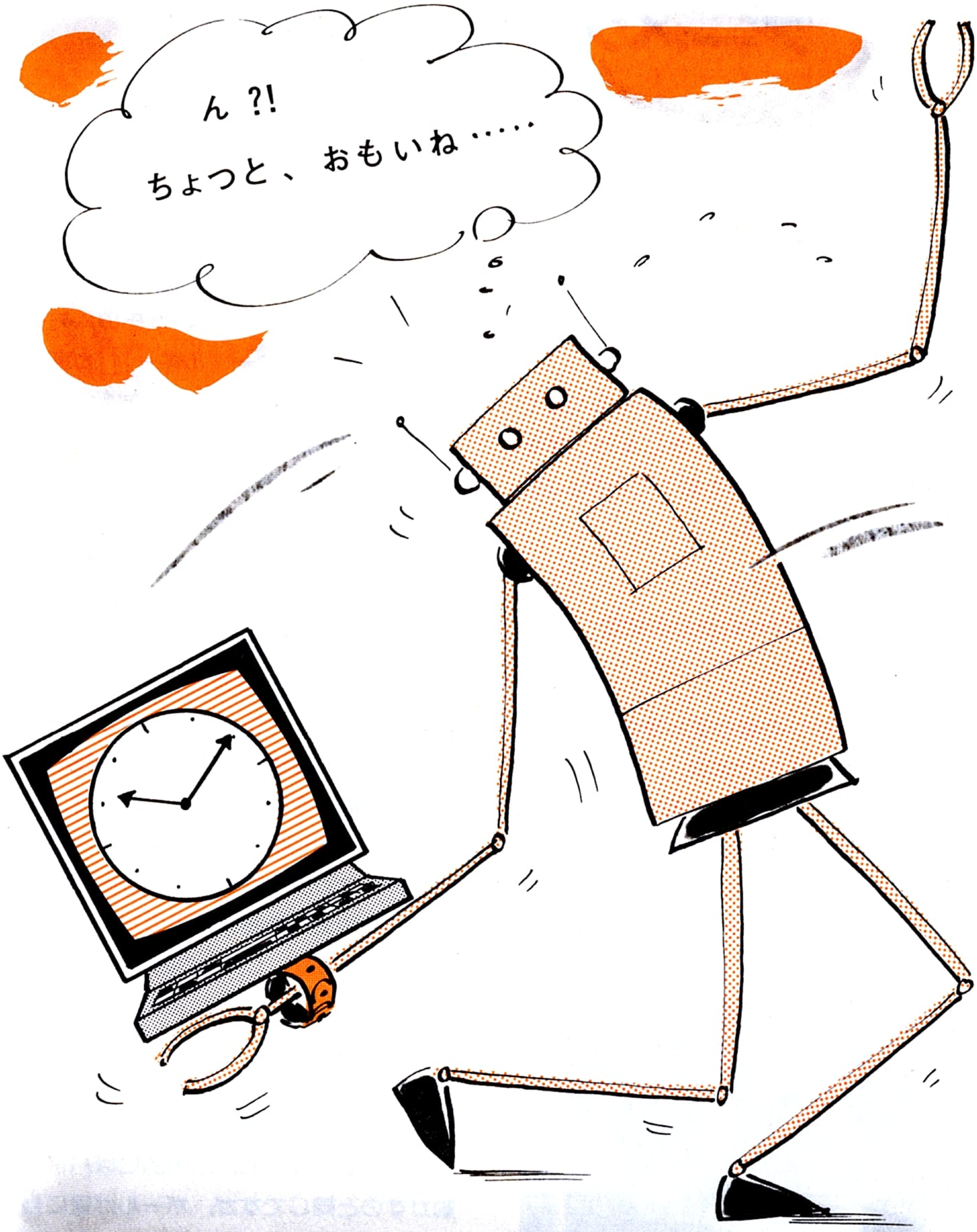
ゲームへのアプローチ



テニスゲームの仕組み
重力の法則
基地が危ない！
スピーディ・スピーディ
図形の拡大・縮小
アナログ時計を作る
迷路を作る
グラフィック・キャラクタで宇宙船を！
五目並べはいかが？
Night Flight
波間にヨットが
イカの編隊飛行

本章では，BASIC プログラムによる TV ゲームの可能性を追求してみたいと思います．

前章では画面に描かれたアートを見て「美しい……」と感心するのが主たる目的でした．しかし単なる静止画面のきれいさだけでは TV ゲームはできません．加えて動きのおもしろさが必要になってくるのです．グラフィックに動きを与えること，これが本章の目的です．


ん ?!
ちょつと、 おもいね……

テニスゲームの仕組み

ゲームの中には，飛行機を上下に動かして敵の砲撃をかわす，ラケットを左右に動かしてボールを打ち返すなど，キーボードからの入力で物を動かす場面が数えきれないほどあります．

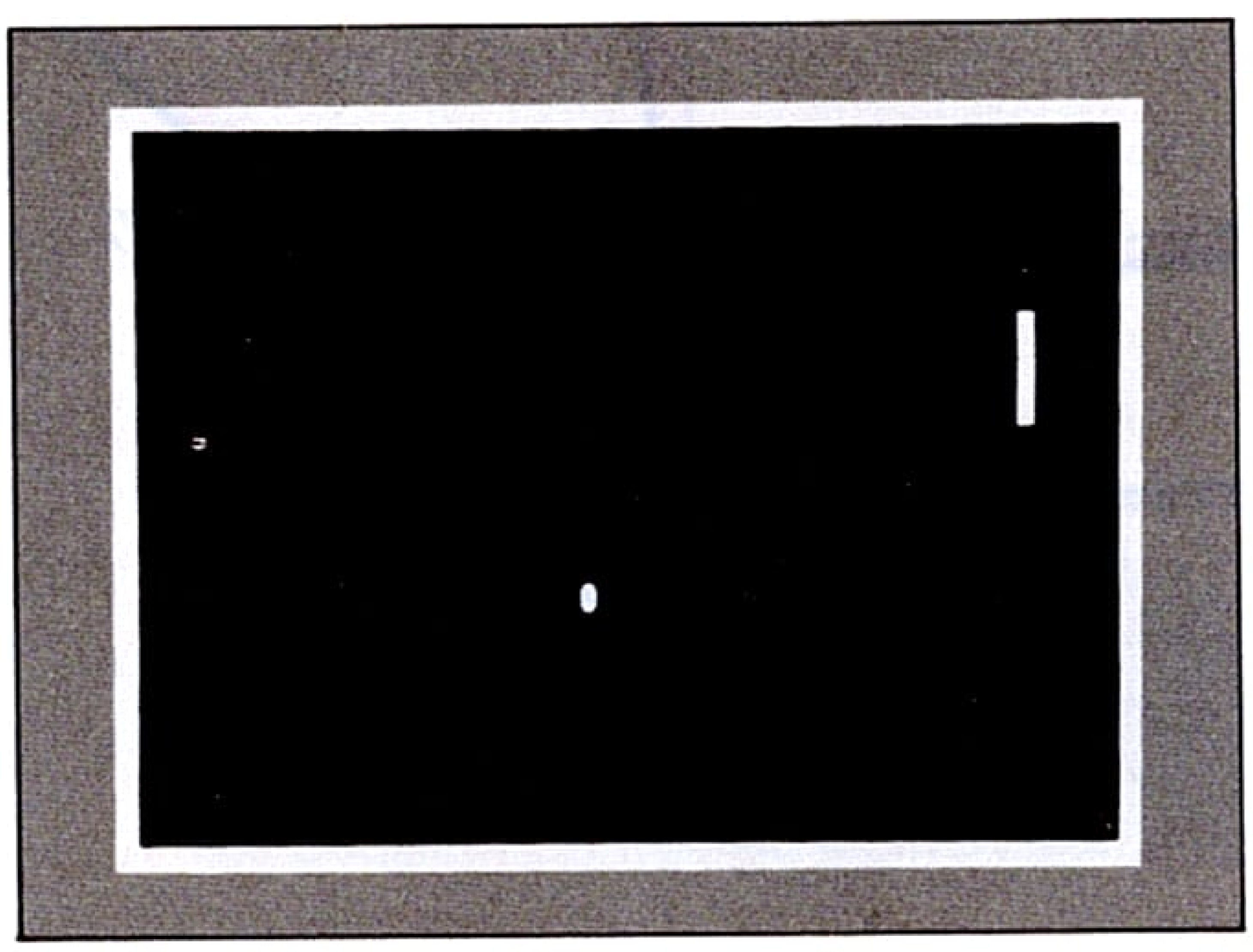
そこで，この「キーボードからの入力で物を動かす」例をお見せしましょう．まずは簡単に，パドルの移動を扱ってみました．

パドルを動かすには，

(1) キーボードからの入力
(2) どんなキーが押されたかの判定
(3) パドルの表示

といったルーチンの繰り返しを行います．

パドルを動かすのではなく，描き直すことに注意してください．画面は動いているように見えても，実際は今描かれているパドルを消し，キー入力によって指定された位置へパドルを再び描き直しています．この



「消しては描く」を高速で行うので，パドルが動いているように見えるのです．

List 4.1a を見てください．210～230行のINP関数で，キーボードからの入力を受けつけます．もしカーソルキー[↑]が押されたら，210行で判定し，パドルを1キャラクタ分上に描き直します．カーソルキー[↓]が押されたら，220行で判定し，パドルは1キャラクタ分下に描き直されます．このプログラムでは，カーソルキーの判定にINP関数を使いましたが，INKEY$やINPUT$を使うことも可能です．

[↑]，[↓]のどちらかが押されると，250行以降のサブルーチンへ分岐します．分岐してパドルを描き直すときに，指定画面外にパドルを描いては困ります．そこで，260，270行に，パドルを描く位置（PY+V）が画面外になるようなら描き直しを行わないという条件文が必要になるのです．実際にパドルを描き直しているのは280，290行です．なお，このプログラムは[ESC]キーにより終了します．

さてこれでパドルを動かすことはできました．次はボールを動かしてみましょう．ボールを動かすのも，原則的にはパドルを動かすのと同じですが，ボールが壁に当たったときの処理については新たに考えなければなりません．そこでボールが壁に当たって反射する仕組みについて考えてみます．

例えば *Figure 1* のようにボールは，右上へ移動して上側の壁に当たると反射して方向が右下へと変わります．また右下へ移動して右側の壁に当たると反射して方向が左下へと変わります．要するに，ボールが当たったと判断したときに，ボールの進行方向を変えればよいのです．

ボールが上下いずれかの壁に当たったときには YS，左右いずれかの壁に当たったときには XS の正負を逆にしてやればボールの進行方向が変わります．

例えば *Figure 2* のように，ボールが

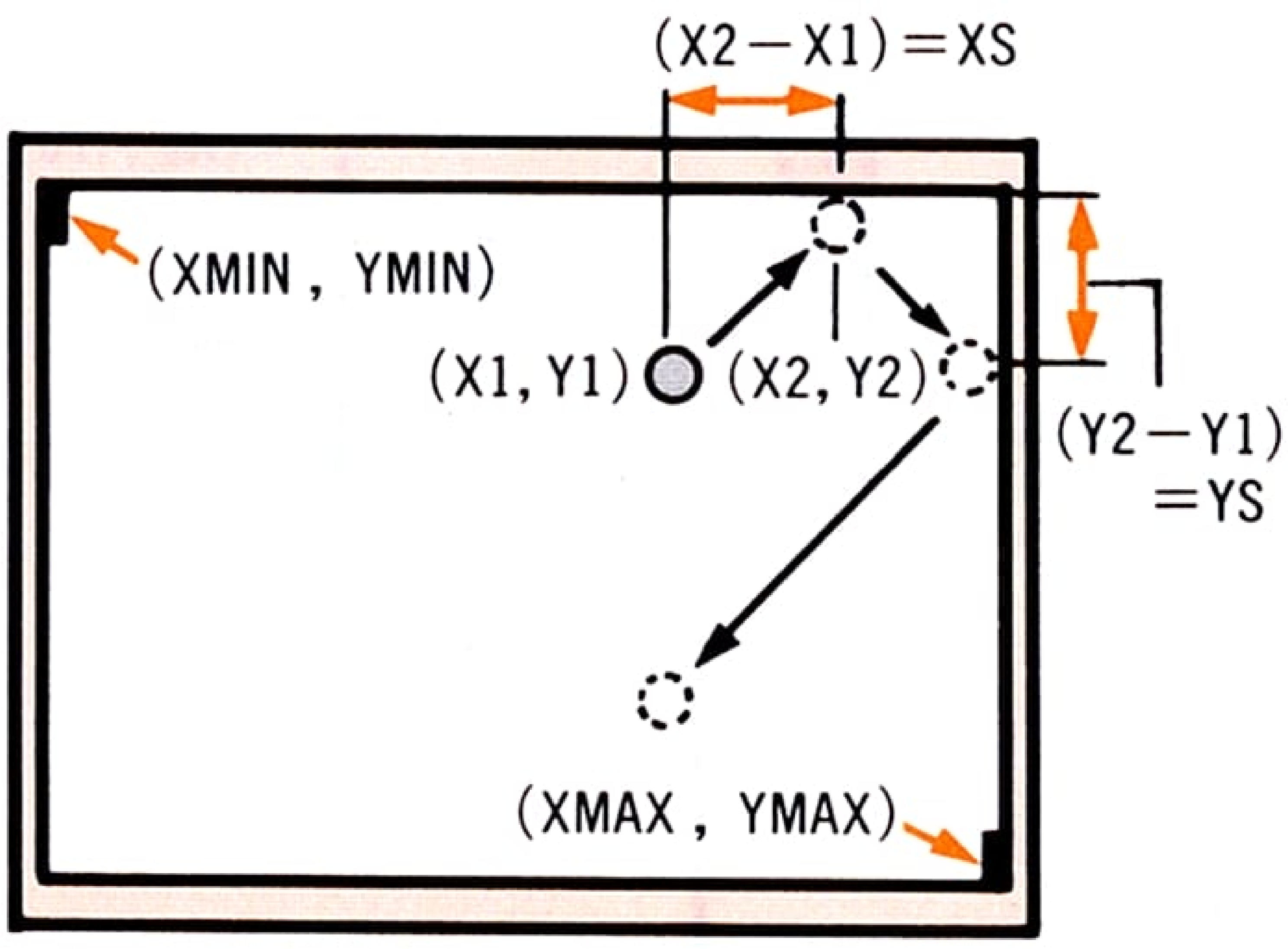


Figure 1

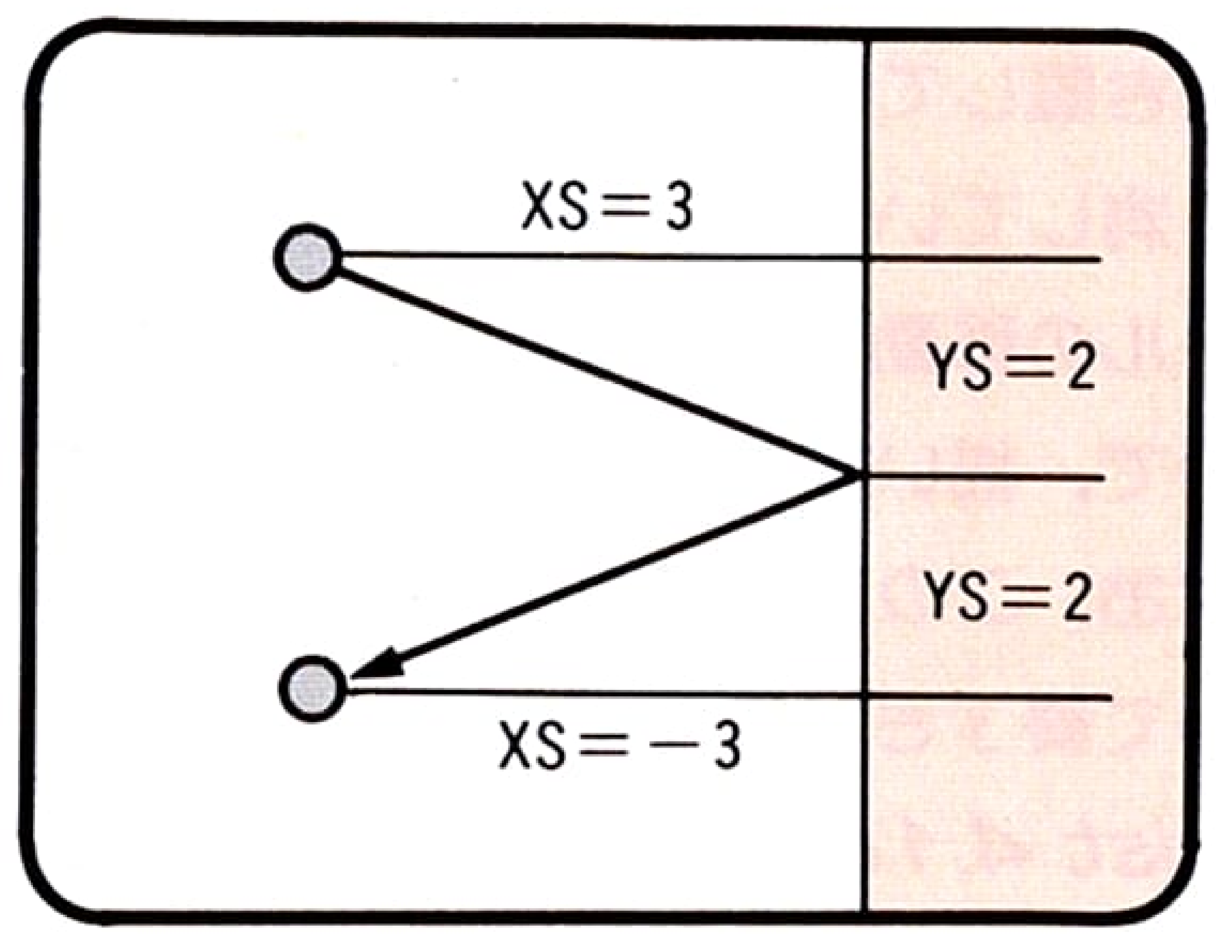


Figure 2

List 4.1a

```
100   List 4.1a
110 '
120 XMAX=68 : YMAX=21 : YMIN=3
130 CL$="  "+CHR$(29)+CHR$(31)+"  "+CHR$(29)+CHR$(31)+"  "
140 PD$="■"+CHR$(29)+CHR$(31)+"■"+CHR$(29)+CHR$(31)+"■"
150 PX=XMAX-5 : PY=(YMAX+YMIN)/2
160 '
170 WIDTH 80,25 : CONSOLE ,,0,1 : SCREEN 0,0 : CLS 3
180 COLOR 7 : OUT 230,0
190 GOSUB *PADDLE
200 *LOOP
210   IF INP(8)<255  THEN V=-1 : GOSUB *PADDLE : GOTO *LOOP
220   IF INP(10)<255 THEN V=1  : GOSUB *PADDLE : GOTO *LOOP
230   IF INP(9)=127  THEN OUT 230,3 : END
240 GOTO *LOOP
250 *PADDLE
260 IF V=-1 AND PY+V<YMIN   THEN RETURN
270 IF V=1  AND PY+V>YMAX-2 THEN RETURN
280 LOCATE PX,PY : PRINT CL$ : PY=PY+V
290 LOCATE PX,PY : PRINT PD$ : RETURN
```

押されたキーの判定（行200～240）

パドルの移動（行250～290）

XS=3, YS=2 で移動して右の壁に当たったときには XS=−3, YS=2 となり，縦方向の進み方は変わりませんが，横方向の進み方を反対にしています．

ここで *List 4.1b* を見てください．壁に当たったときの処理は，210，220 行で行っており，ボールを描き直しているのは，240，250 行です．パドルのときは消してから描き直していましたが，ボールは描いてから消しています．パドルの移動量に比べ，ボールの移動量が大きく，ちらつきが目立つので，描いてから消しているのです．

なお，このプログラムは，ESC キーを押して終了させてください．

List 4.1a, *List 4.1b* を利用して，ボールをパドルで受けられるようにしてみました．*List 4.1c* を見てください．この 330 行は，ボールがパドルに当たったとき反射するためのものです．ボールが進む方向にパドルがあるかないかを判定して，パドルがあった場合は，XS の正負を逆にしてい

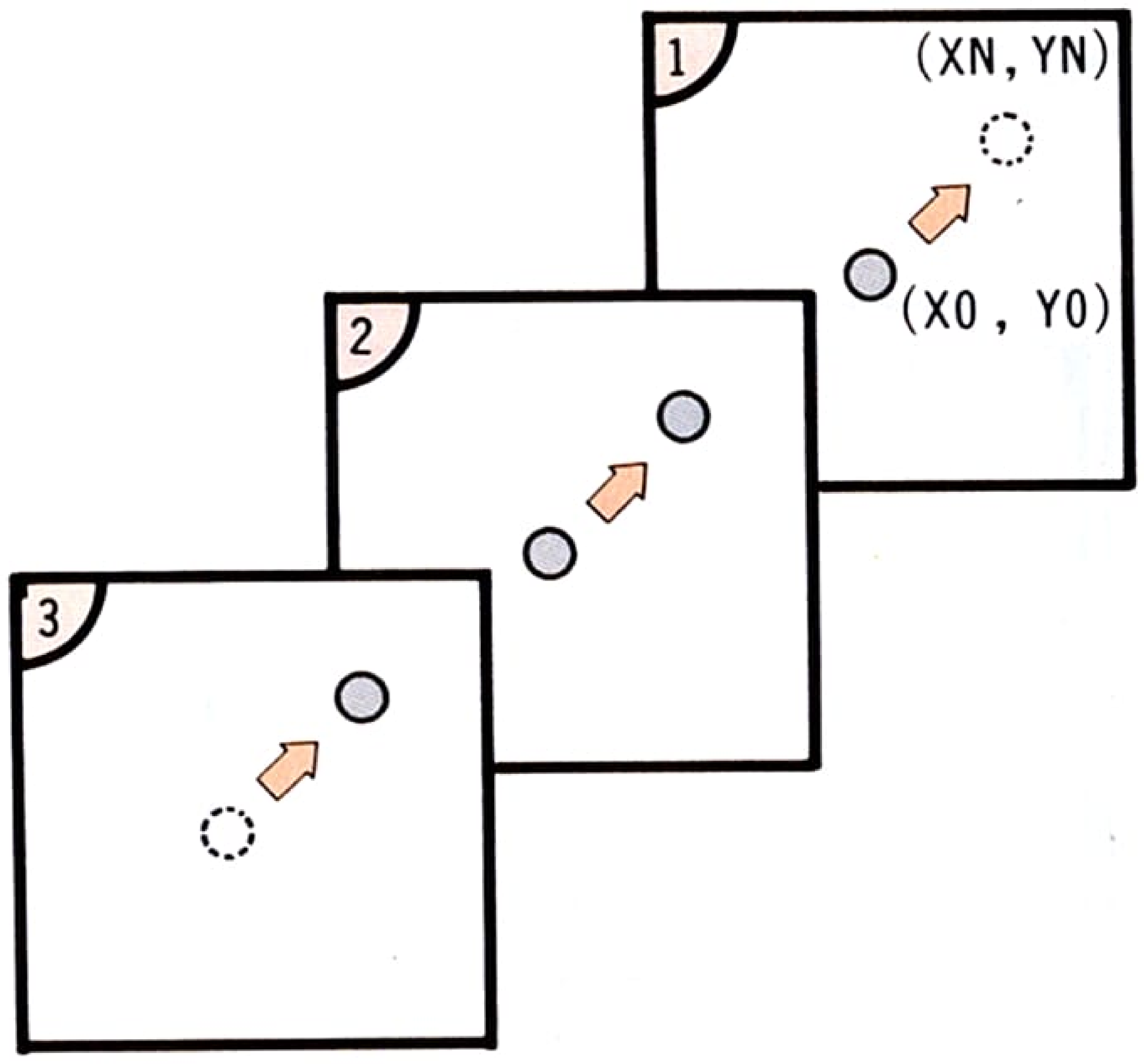


Figure 3

List 4.1b

```
100 ' List 4.1b
110 '
120 XMAX=68 : YMAX=21 : XMIN=13 : YMIN=3 : XS=-2 : YS=1      ←ボールの初速度
130 XO=INT((XMIN+XMAX)/2) : YO=INT((YMIN+YMAX)/2)
140 '
150 WIDTH 80,25 : CONSOLE ,,0,1 : SCREEN 0,0 : CLS 3 : COLOR 7
160 '
170 LINE (XMIN*8-1,YMIN*8-1)-(XMAX*8+8,YMAX*8+8),7,B
180 LINE (XMIN*8-9,YMIN*8-5)-(XMAX*8+16,YMAX*8+12),7,B
190 PAINT (XMIN*8-2,YMIN*8-2),7
200 *BALL
210 IF XO+XS>XMAX OR XO+XS<XMIN THEN XS=-XS  } 壁に当たったときの処理
220 IF YO+YS>YMAX OR YO+YS<YMIN THEN YS=-YS  }
230 XN=XO+XS : YN=YO+YS
240 LOCATE XN,YN : PRINT "●";  } ボールの移動
250 LOCATE XO,YO : PRINT " ";  }
260 XO=XN : YO=YN
270 IF INP(9)=127 THEN OUT 230,3 : END
280 GOTO *BALL
```

ます．これは，壁でボールを反射させるときとまったく同じです．

なお，XS，YS を変えると，ボールの速さや方向を変えられます．

List 4.1c

```
100 ' List 4.1c
110 '
120 XMAX=68 : YMAX=21
130 XMIN=13 : YMIN=3
140 XS=-2   : YS=1
150 CL$=" "+CHR$(29)+CHR$(31)+" "+CHR$(29)+CHR$(31)+" "
160 PD$="■"+CHR$(29)+CHR$(31)+"■"+CHR$(29)+CHR$(31)+"■"
170 PX=XMAX-5 : PY=(YMAX+YMIN)/2
180 XO=INT((XMIN+XMAX)/2) : YO=INT((YMIN+YMAX)/2)
190 '
200 WIDTH 80,25 : CONSOLE ,,0,1 : SCREEN 0,0 : CLS 3
210 COLOR 7 : OUT 230,0
220 '
230 LINE (XMIN*8-1,YMIN*8-1)-(XMAX*8+8,YMAX*8+8),7,B
240 LINE (XMIN*8-9,YMIN*8-5)-(XMAX*8+16,YMAX*8+12),7,B
250 PAINT (XO*8,YO*8),CHR$(0)+CHR$(0)+CHR$(85),7
260 PAINT (0,0),CHR$(0)+CHR$(85)+CHR$(0),7
270 PAINT (XMIN*8-2,YMIN*8-2),7
280 GOSUB *PADDLE
290 '
300 *LOOP
310 IF XO+XS>XMAX OR XO+XS<XMIN THEN XS=-XS
320 IF YO+YS>YMAX OR YO+YS<YMIN THEN YS=-YS
330 IF XO+XS>PX-ABS(XS) AND XO+XS<PX+ABS(XS) AND YO+YS>=PY AND
               YO+YS<=PY+2 THEN BEEP 1 : BEEP 0 : XS=-XS
340 XN=XO+XS : YN=YO+YS
350 LOCATE XN,YN : PRINT "●";
360 LOCATE XO,YO : PRINT " ";
370 XO=XN : YO=YN
380 '
390   IF INP(8)<255  THEN V=-1 : GOSUB *PADDLE : GOTO *LOOP
400   IF INP(10)<255 THEN V=1  : GOSUB *PADDLE : GOTO *LOOP
410   IF INP(9)=127  THEN OUT 230,3 : END
420 GOTO *LOOP
430 '
440 *PADDLE
450 IF V=-1 AND PY+V<YMIN   THEN RETURN
460 IF V=1  AND PY+V>YMAX-2 THEN RETURN
470 LOCATE PX,PY : PRINT CL$
480 PY=PY+V
490 LOCATE PX,PY : PRINT PD$
500 RETURN
```

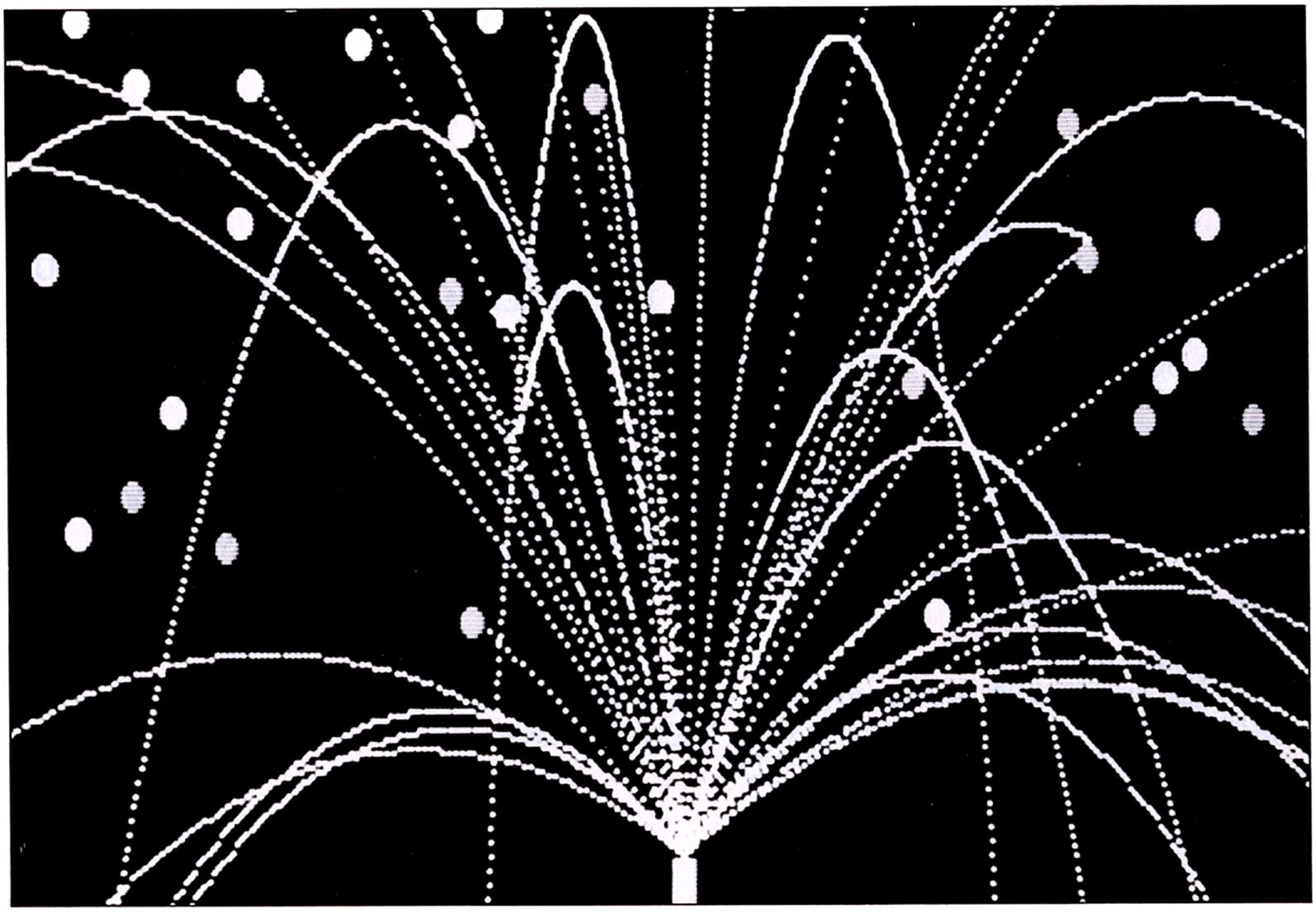
パドルにボールが当たったときの処理

重力の法則

インベーダーにミサイルが当たったり，敵の基地を破壊したり，ゲームでは「何かが何かに当たる」という状況が，非常に多く見受けられるものです．プログラムでは，どうやって当たりはずれを判断しているのでしょうか．ここでは，発射台から打ち上げたボールが物体めがけて飛んで行く場合を考えてみました．

物体の数が少なければ，ボールの位置と物体の距離を，そのつど計算してみることもできます．しかし，物体が1万個もあったりすると，ボールとの距離を調べているだけで，日が暮れてしまいます．そこで，何かがあるかないかを座標の位置から算出するのではなく，直接画面上のドットを調べて判断するために，POINT 関数が用意されています．

POINT（X，Y）は，もしグラフィック

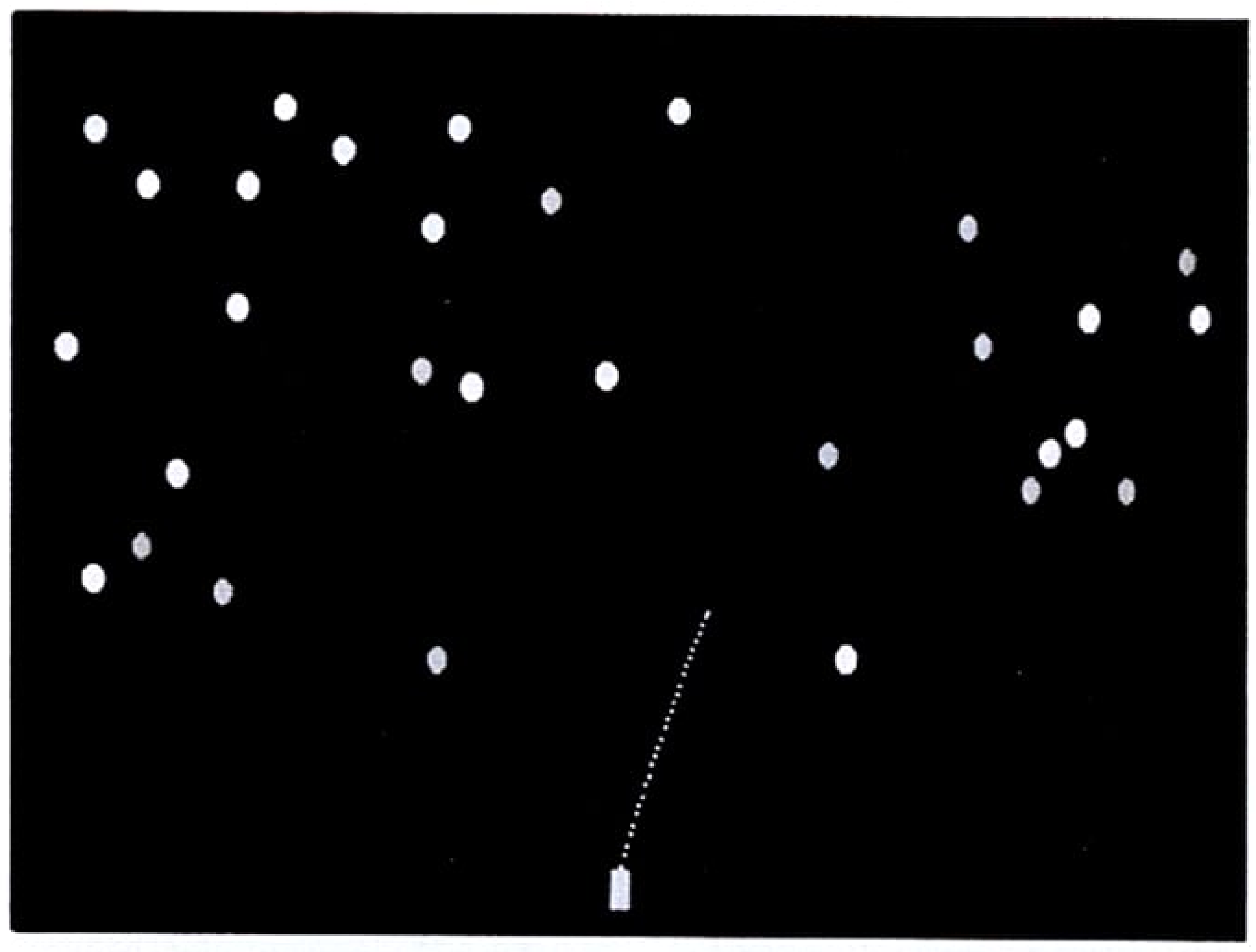


座標(X，Y)に何か書いてあればそのパレットコードを，何もなければ0を返してきます．そこで，POINT関数で1つ先のドットを調べて，何もなければボールを1ドット進め，何かあれば命中したとみなすというわけです．

　ここでプログラムを見てみましょう．140〜170行で的となる物体をランダムにちりばめています．250〜320行のループで，1ドット先をPOINT関数で調べながら進んでいきます．ボールが的にあたるとBEEP音がするという処理を280行でしています．もしボールが画面の外に出てしまったら，的には当たらなかったことになります．310行では，何かキーを押すとプログラムは終了するという処理を行っています．

　なお，このプログラムでは，ボールの動きを重力の存在を考えた上で決定しています．その計算は260行で行っています．

　噴水みたいでなかなかきれいでしょう?



List 4.2

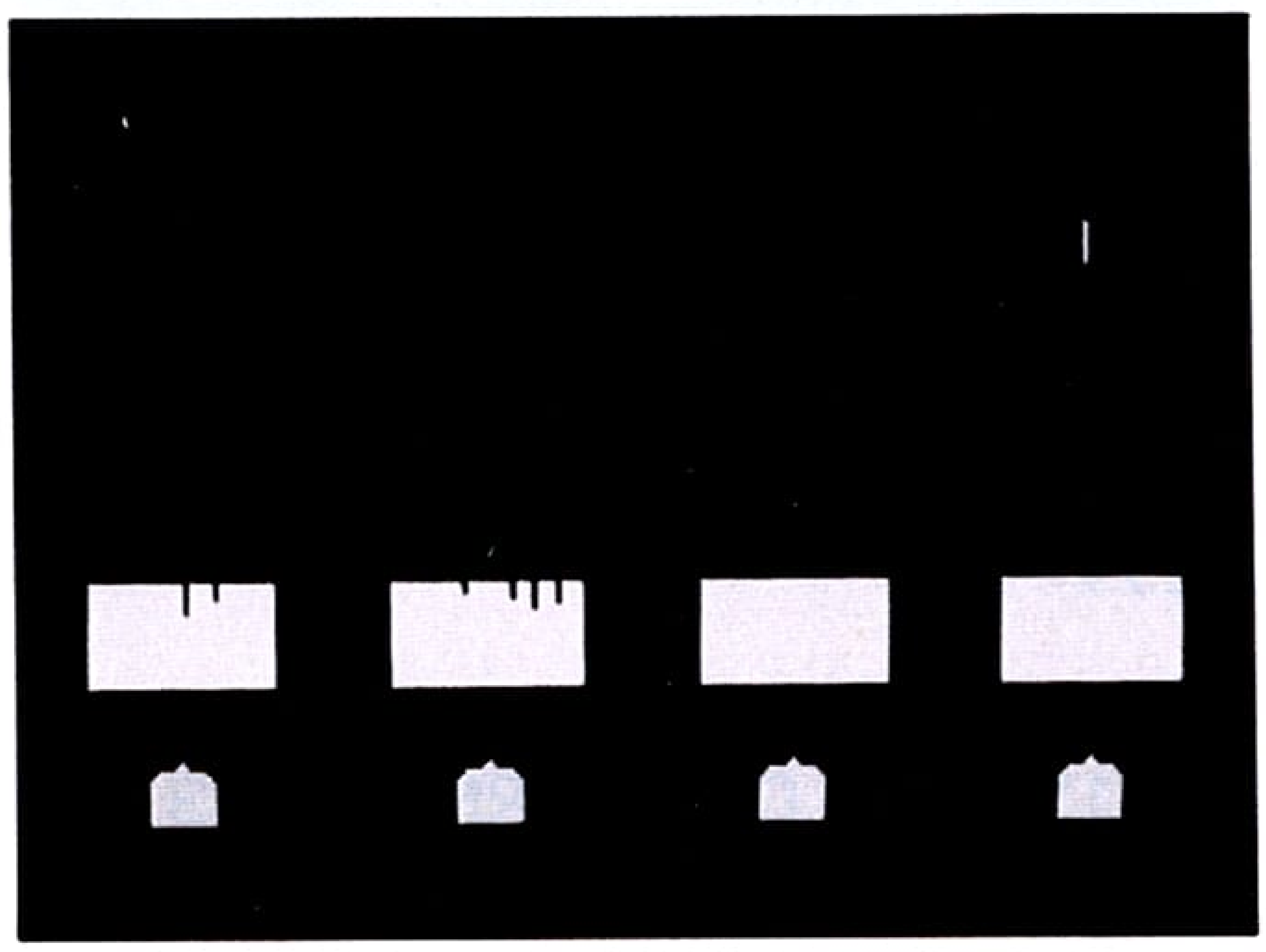
```
100 ' List 4.2
110 '
120 RANDOMIZE VAL(RIGHT$(TIME$,2))
130 SCREEN 0,0 : CLS 3
140 FOR I=1 TO 30
150   X=INT(RND*640) : Y=INT(RND*150) : C=INT(RND*3)+3  } 的をランダム
160   CIRCLE (X,Y),5,C : PAINT (X,Y),C                   } に表示
170 NEXT I
180 LINE (315,190)-(325,199),2,BF
190 '
200 *INIT
210 V=(INT(RND*6)+5)*10
220 TH=(RND*140+20)*3.14159/180 : VX=V*COS(TH) : VY=V*SIN(TH)
230 T=0
240 '
250 *LOOP
260 X=VX*T+320 : Y=189-(VY*T-4.9*T*T)……ボールの位置の計算
270 IF X<0 OR X>639 OR Y<0 OR Y>199 GOTO *INIT
280 IF POINT(X,Y)>0 AND POINT(X,Y)<7 THEN BEEP : GOTO *INIT
290 BEEP 1 : BEEP 0 : PSET (X,Y),7
300 T=T+.05
310 IF INKEY$<>"" THEN END
320 GOTO *LOOP
```

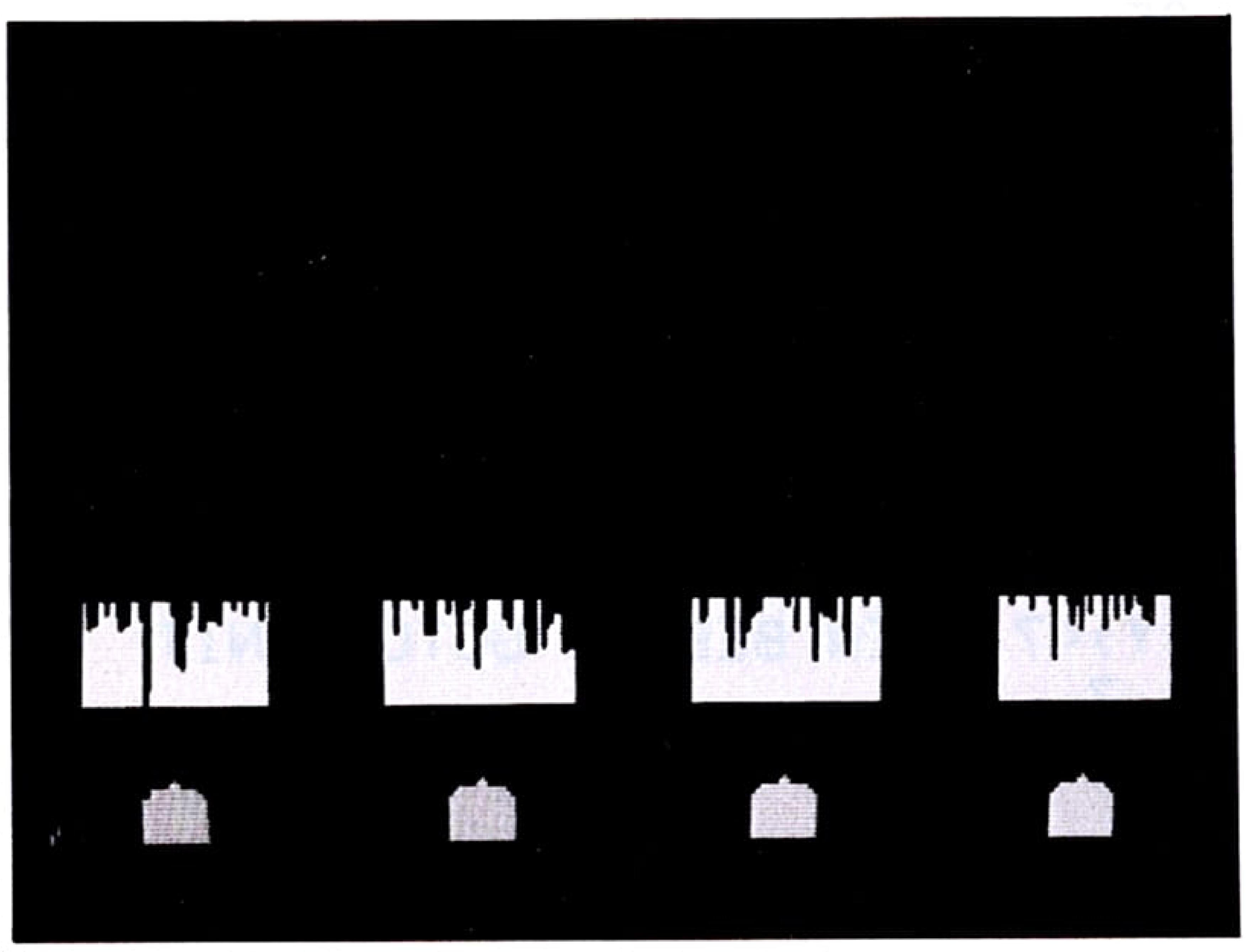
基地が危ない！

降りそそぐミサイル．防御用の壁が，目の前でみるみる破壊されていく．ミサイル基地が危ない！ なんて場面がゲームにはよくありますが，いったいどうしたらこんな画面が描けるのでしょうか．シンプルなかたちでそれを再現してみたのがこのプログラムです．

ミサイルは，その先に物体があるかないかを調べながら降りそそいでいます．その判定には，やはりPOINT関数を使っています．このミサイルは，1度に縦8ドットずつ落ちていきます．そして，壁にミサイルが当たったときにくい込んだ分だけ壁は破壊されていきます（*Figure 1* 参照）．この破壊されてできた細長い穴がだんだん重なって，ついには壁を貫き，基地にミサイルが命中してしまうわけです．

まず，壁と基地を150～210行で描いていきます．この中で，LINE－STEPという命令がありますが，これは現在の位置から





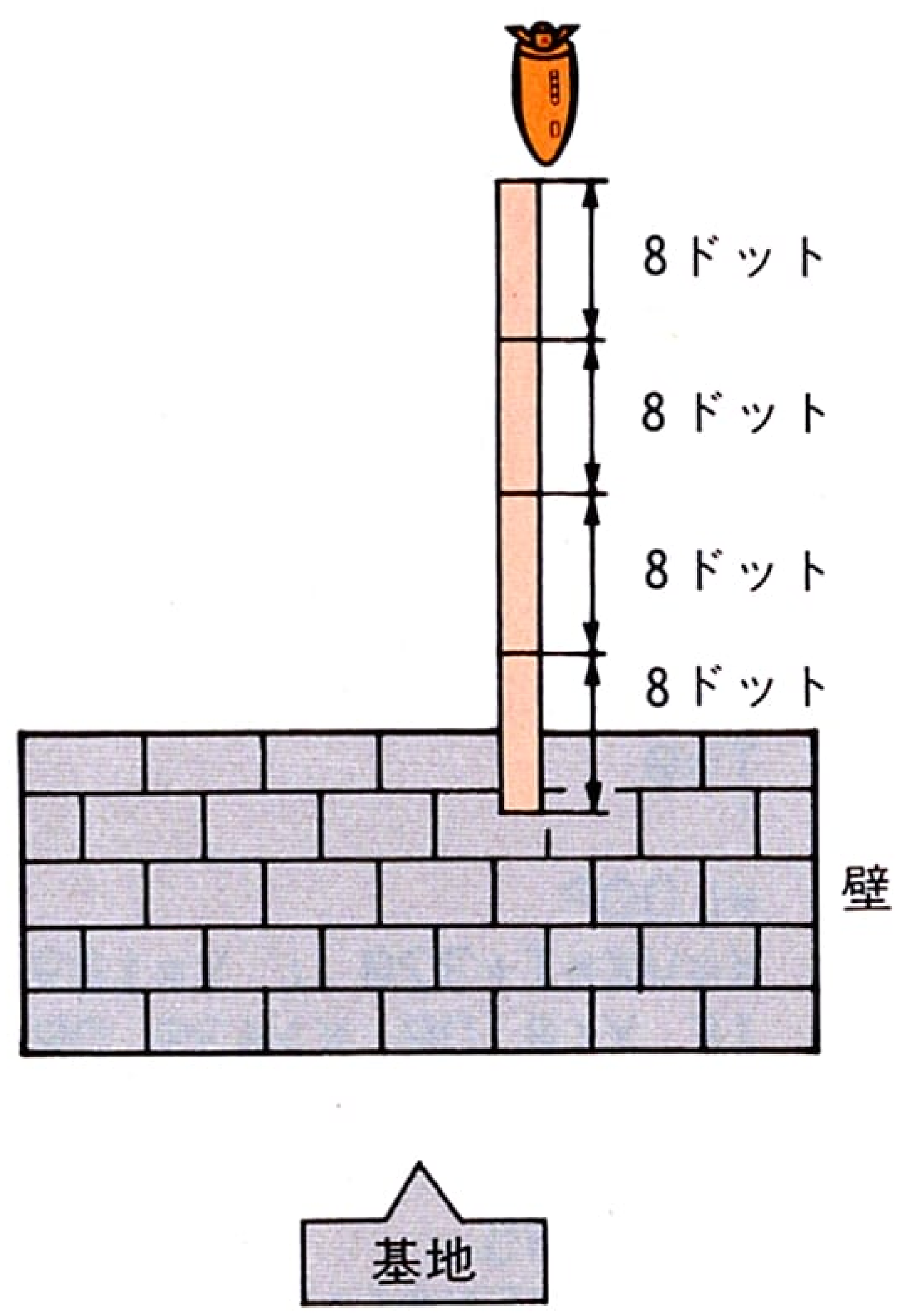


Figure 1

横へ何ドット，縦へ何ドット先のところまで線を引けという命令で，相対座標と呼ばれるものです．この相対座標を使うと，基準点を1つ決めてしまえば，同じ図形を異なる場所にいくつも描くことができます．160行と170行のPOINT命令で，それぞれ壁と基地を描くときの基準点を決めています．次に，ミサイルは，250～320行のループで動かしています．このときミサイルの行き先をPOINT関数で調べ，水色ならば基地に命中したとして340行のBEEP音を出力するルーチンへ飛びます．

なお，ミサイルは，290，300行のLINE命令で描いては消しています．ゲームらしさを出したければ，ミサイルの代わりに，何か自分なりのグラフィック・キャラクタを作って，それを動かすサブルーチンを考えてみるとよいでしょう．

List 4.3

```
100 ' List 4.3
110 '
120 MX=1 : MY=8 : MC=6 : RANDOMIZE VAL(RIGHT$(TIME$,2))
130 '
140 SCREEN 0,0 : CLS 3
150 FOR I=1 TO 4
160  POINT (I*130,180) : LINE STEP(-40,-35)-STEP(80,20),7,BF
170  POINT (I*130,180) :LINE -STEP(3,2),5 :LINE -STEP(7,0),5
180  LINE -STEP(4,2),5 :LINE -STEP(0,8),5 :LINE -STEP(-28,0),5
190  LINE -STEP(0,-8),5:LINE -STEP(4,-2),5:LINE -STEP(7,0),5
200  LINE -STEP(3,-2),5: PAINT STEP(0,5),5
210 NEXT I
220 '
230 *START : X=INT(RND*540)+50 : Y=INT(RND*6)+10
240 '
250 *LOOP
260   IF POINT(X,Y+1)=7 OR Y+MY>199 GOTO *START
270   IF POINT(X,Y+1)=5 GOTO *EFFECT
280   Y=Y+MY
290   LINE (X-MX,Y)-(X+MX,Y-MY),MC,BF
300   LINE (X-MX,Y)-(X+MX,Y-MY),0,BF
310   IF INKEY$<>"" THEN END
320 GOTO *LOOP
330 '
340 *EFFECT
350 FOR I=1 TO 10
360   BEEP 1 : FOR J=1 TO I*15 : NEXT J : BEEP 0
370 NEXT I
```

壁と基地を描く（160～200行）

ミサイルを動かす（250～320行）

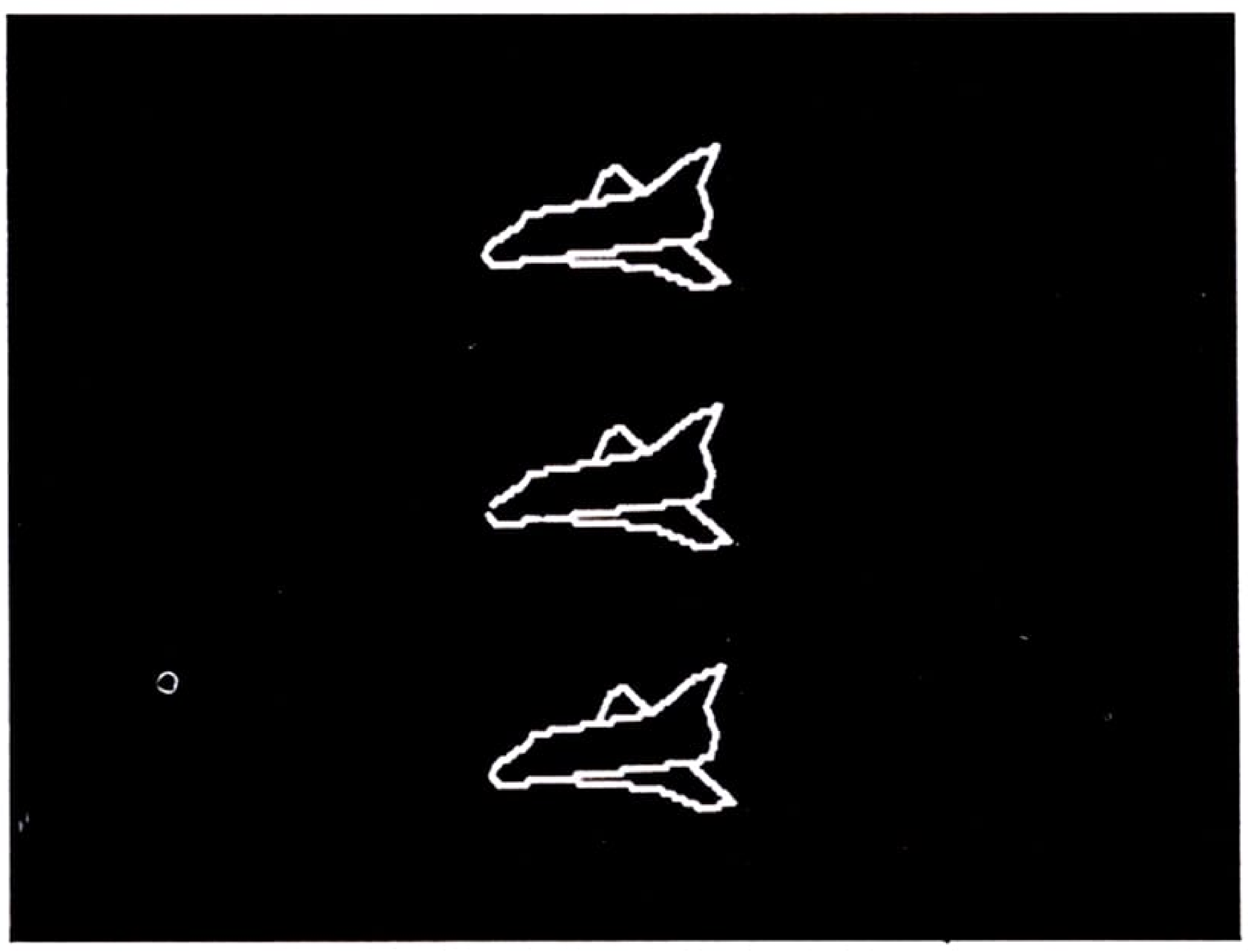
命中したときの処理（340～370行）

スピーディ・スピーディ

リアルタイムなシューティングゲームを作ろうと思っても，BASIC じゃ遅くっておもしろいゲームなんてできない，そう思ってはいませんか？ そこで，スピード感をパレット機能で補う方法を紹介しましょう．今までは，物を動かすのに描いては消す，もしくは消してから描くという方法を用いてきました．ここではそれをもっと速く動くように見せるために，初めから描いておき，必要なものだけ見せるという方法を用います．

例えば，*Figure 1* のようにあらかじめ動かす方向にボールを描いておきます．この場合，ボールの色を動く方向に対応させて，パレットコードで，1，2，3，……6と並べる必要があります．

描き終わった時点で，パレット機能によりすべての色に黒を割り当てて１度消します．同様にパレット機能で，パレットコード１(始めに青で描いたもの)のみを白に変えてボールを表示し，すぐにパレットコード１に黒を割り当ててボールを消します．次にパレットコード２のみを白に変えて表示する，さらに紫を……というように，順



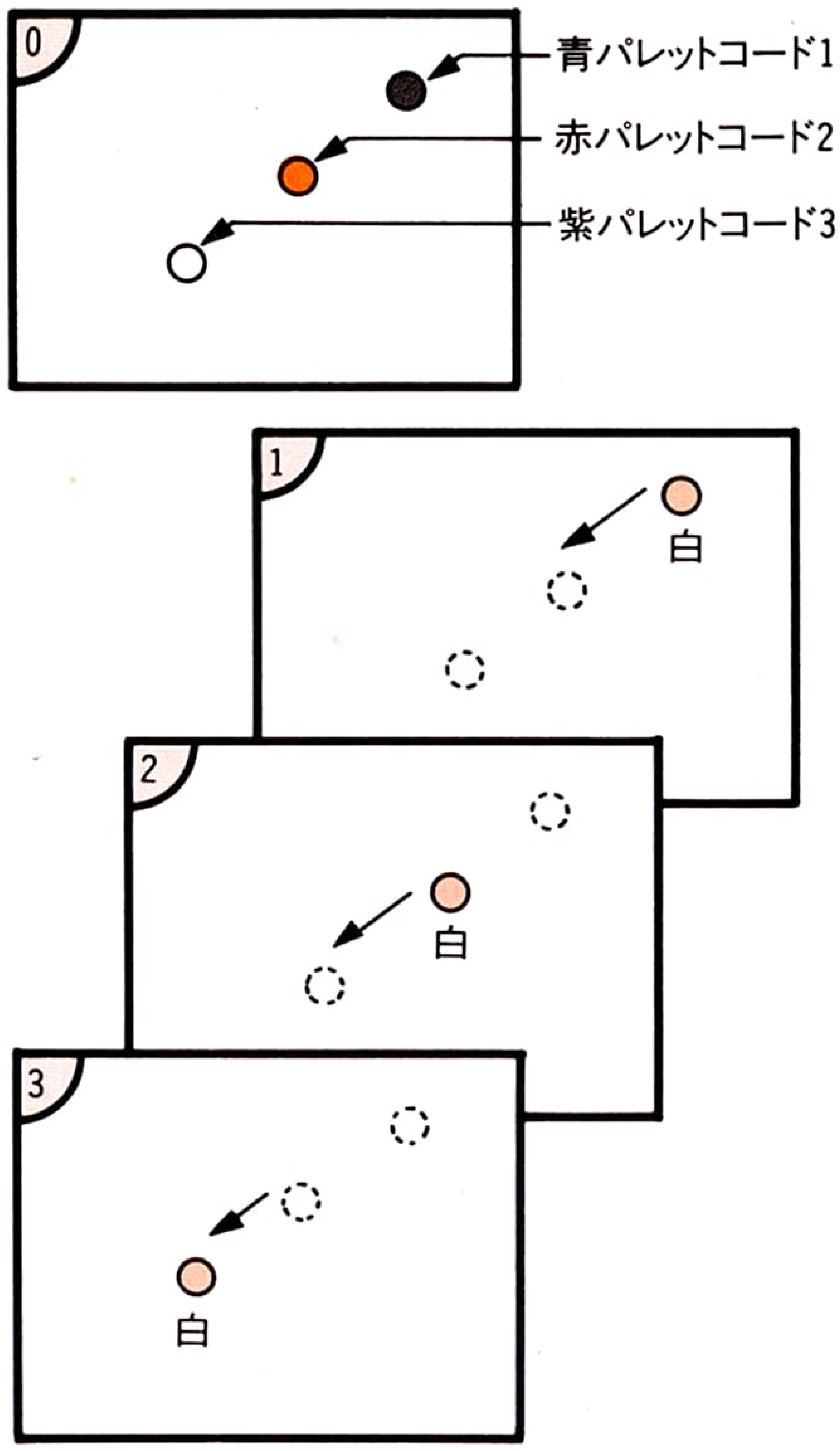


Figure 1

次表示させていけば，ボールは高速で移動しているように見えるのです(*Figure 2* 参照).

ただし，パレットによって動かせるのは7動作までです．PC-8801 mk IIの使えるカラーは8色ありますが，バックグラウンドカラーに1色（通常は黒）使うので残りは7色です．あらかじめ描いておくボールには，残りの7色を使うわけですから，動作としては7動作までになります．

それでは *List 4.4a* を見てください．160〜210行のFOR〜NEXTループで，青，赤，紫，緑，水，黄，白の7色を使ってあらかじめスペースシャトルを描いてお

List 4.4a

```
100 ' List 4.4a
110 '
120 XMAX=639 : XS=-(XMAX-150)/7 : YS=26
130
140 SCREEN 0,0 : CLS 3
150 C=1
160 FOR I=0 TO 6
170   X=XMAX-150+XS*I : Y=0+YS*I : GOSUB *SPACE
180   X=XMAX-150+XS*I : Y=0+YS*3 : GOSUB *SPACE
190   X=XMAX-150+XS*I : Y=0+YS*(6-I) : GOSUB *SPACE
200   C=C+1
210 NEXT I
220 '
230 *MAIN
240 FOR I=1 TO 7 : COLOR=(I,0) : NEXT I
250 FOR I=1 TO 7
260   COLOR=(I,7)
270   FOR J=1 TO 100 : BEEP 1 : BEEP 0 : NEXT J
280   COLOR=(I,0)
290 NEXT I
300 FOR I=1 TO 7 : COLOR=(I,I) : NEXT I
310 END
320 '
330 *SPACE : RESTORE 410
340 FOR J=1 TO 25
350   IF J=1 OR J=22 OR J=24 THEN F=-1
360   READ A,B : GX=X+A : GY=Y+B
370   IF F=-1 THEN F=0 : POINT (GX,GY)
380   IF F=0  THEN LINE -(GX,GY),C
390 NEXT J : RETURN
400 '
410 DATA  70, 9,  90, 2, 102, 0,  94, 8,  98,13,  96,18,  86,19
420 DATA 106,27,  92,28,  72,24,  26,22,   4,23,   0,21,   2,19
430 DATA  16,15,  18,13,  46,11,  48, 9,  52, 5,  58, 4,  70, 9
440 DATA  46,11,  70, 9,  26,22,  86,19
```

160〜210行: あらかじめスペースシャトルを描いておくルーチン

250〜290行: パレットコードを換え，スペースシャトルを動かす

330〜390行: 1機のスペースシャトルを描くサブルーチン

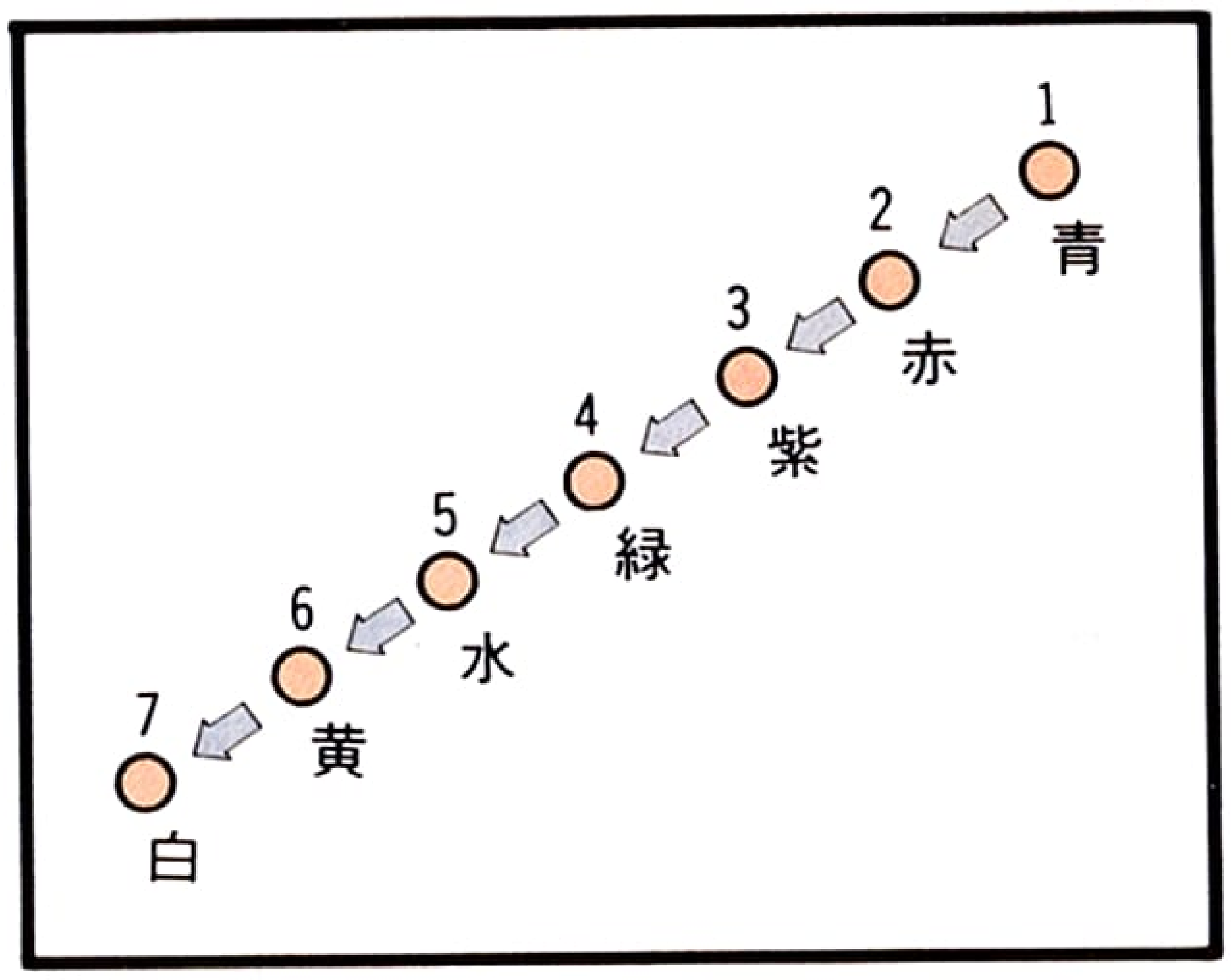


Figure 2

きます．240 行ではこの 7 色すべてを黒に置き換え，シャトルを 1 度消します．そして，次の 250～290 行でシャトルを動かしています．

250～290 行の FOR～NEXT ループでパレットコード 1（通常は青）に白を割り当て，1 番目のシャトルを表示します．次にパレットコード 1 に黒を割り当てて，シャトルを消し，次のパレットコード 2 に白を割り当てて，2 番目のシャトルを表示する……という操作を，パレットコード 7 まで繰り返します．そして 300 行でパレットを初期化してこのプログラムを終了するわけです．

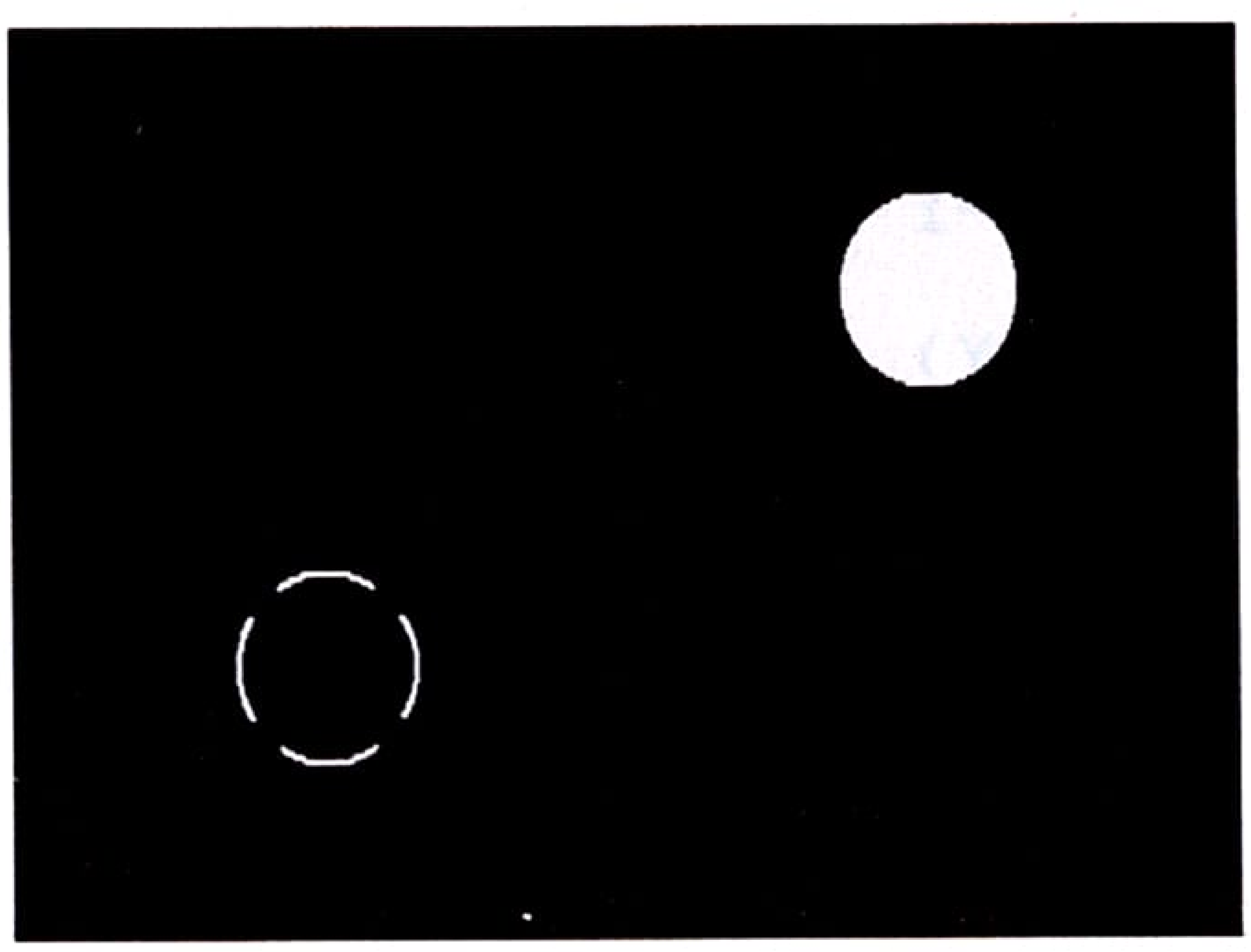
このプログラムのようにパレットを操作する場合，プログラムの途中でストップすると，パレットコードに対応する色が，バラバラになってしまいます．そのようなときは，必ずパレットコードを初期化してください．

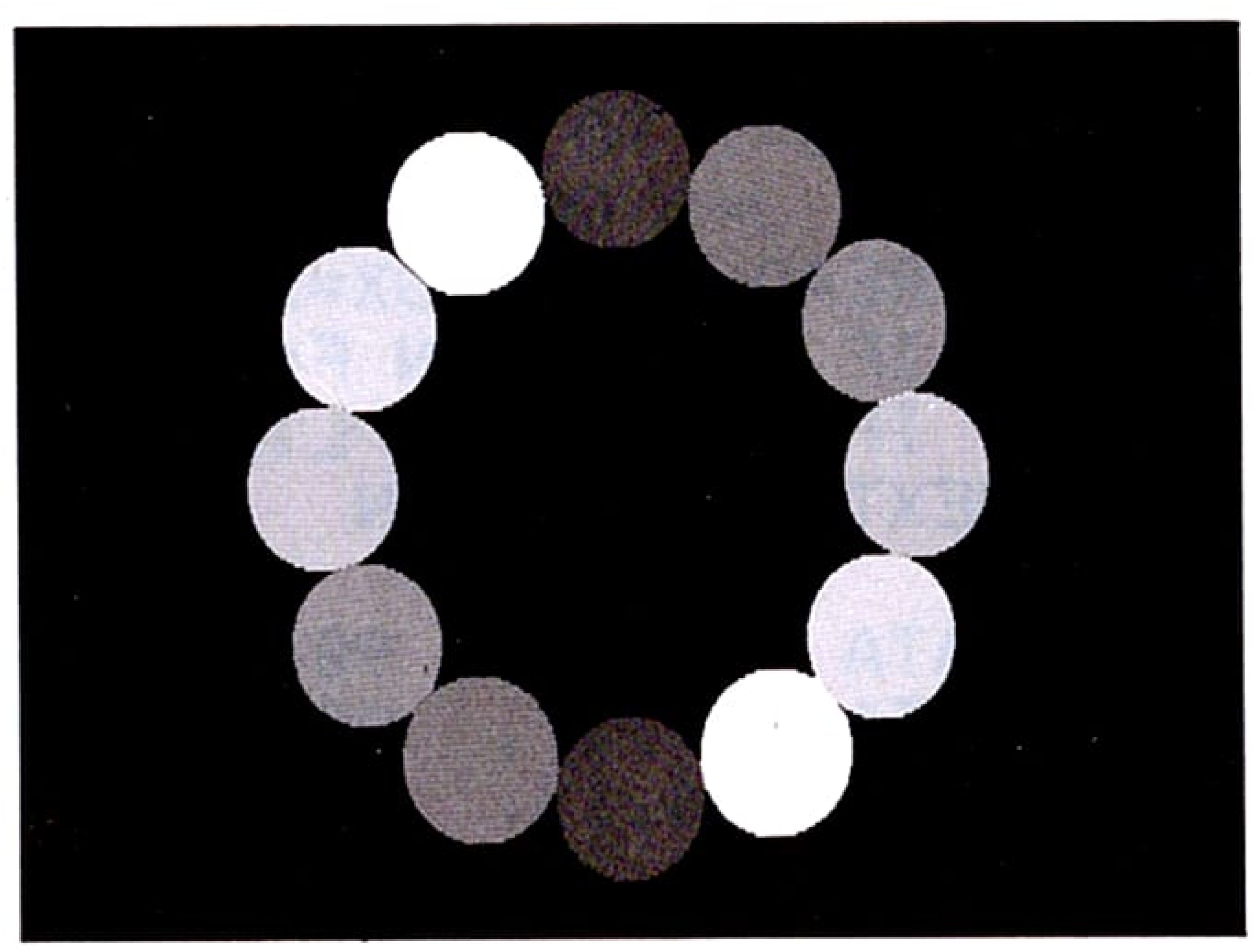
それでは，次に 2 つの円が，ある 1 点を中心に回転する場合を考えます．「円を回転させる▷円を移動する▷円を描いて消す，の繰り返し，さぁプログラムを組もう」ということでできたのは，*List 4.4b* です．これを実行してみてください．

回転する円がたいへんなスピードで動き，あまりの感激にあくびが止まらなかったのではないでしょうか？

ここで，パレット機能を使ってみましょう．*List 4.4c* を打ち込んでみてください．あらかじめ書き込んだものを単につけたり消したりしているだけで，当然といえば当然なのですが，十分利用できるスピードでしょう．なお，このプログラムは何かキーが押されると終了します．

この速さを活かして，ルーレットでも作ってみますか？





List 4.4b

```
100 ' List 4.4b
110 '
120 P=.5 : PI=3.14159
130 X0=320 : Y0=100 : R=150 : PC=5
140 '
150 SCREEN 0,0 : CLS 3 : C=1
160 '
170 *LOOP
180 FOR TH=0 TO PI*2 STEP PI/6
190   X1=R*COS(TH)    : Y1=-R*P*SIN(TH)
200   X2=R*COS(TH+PI) : Y2=-R*P*SIN(TH+PI)
210   C=2 : X=X1 : Y=Y1 : GOSUB *DRAW
220   C=2 : X=X2 : Y=Y2 : GOSUB *DRAW
230   C=0 : X=X1 : Y=Y1 : GOSUB *DRAW
240   C=0 : X=X2 : Y=Y2 : GOSUB *DRAW
250   IF INKEY$>"" THEN END
260 NEXT TH
270 END
280 '
290 *DRAW
300 CIRCLE (X0+X,Y0+Y),R/4,C : PAINT (X0+X,Y0+Y),C,C : RETURN
```

180〜200: 円を描く位置を計算する
210〜220: 円を描く
230〜240: 円を消す

List 4.4c

```
100 ' List 4.4c
110 '
120 P=.5 : PI=3.14159
130 X0=320 : Y0=100 : R=150 : PC=2
140 '
150 SCREEN 0,0 : CLS 3 : C=1
160 FOR TH=0 TO 2*PI STEP PI/6
170   X=R*COS(TH) : Y=-R*P*SIN(TH)
180   CIRCLE (X0+X,Y0+Y),R/4,C : PAINT (X0+X,Y0+Y),C
190   C=C+1 : IF C=7 THEN C=1
200 NEXT TH
210 FOR I=1 TO 6 : COLOR=(I,0) : NEXT I
220 '
230 *LOOP
240 FOR I=1 TO 6
250   COLOR=(I,PC)
260   FOR J=1 TO 30 : NEXT J
270   COLOR=(I,0)
280 NEXT I
290 IF INKEY$="" GOTO *LOOP
300 FOR I=1 TO 6 : COLOR=(I,I) : NEXT I : END
```

160〜200: あらかじめ円を描いておく
230〜280: パレットコードを変えて円を移動させる
290〜300: 何かキーを押すとプログラムを終了する

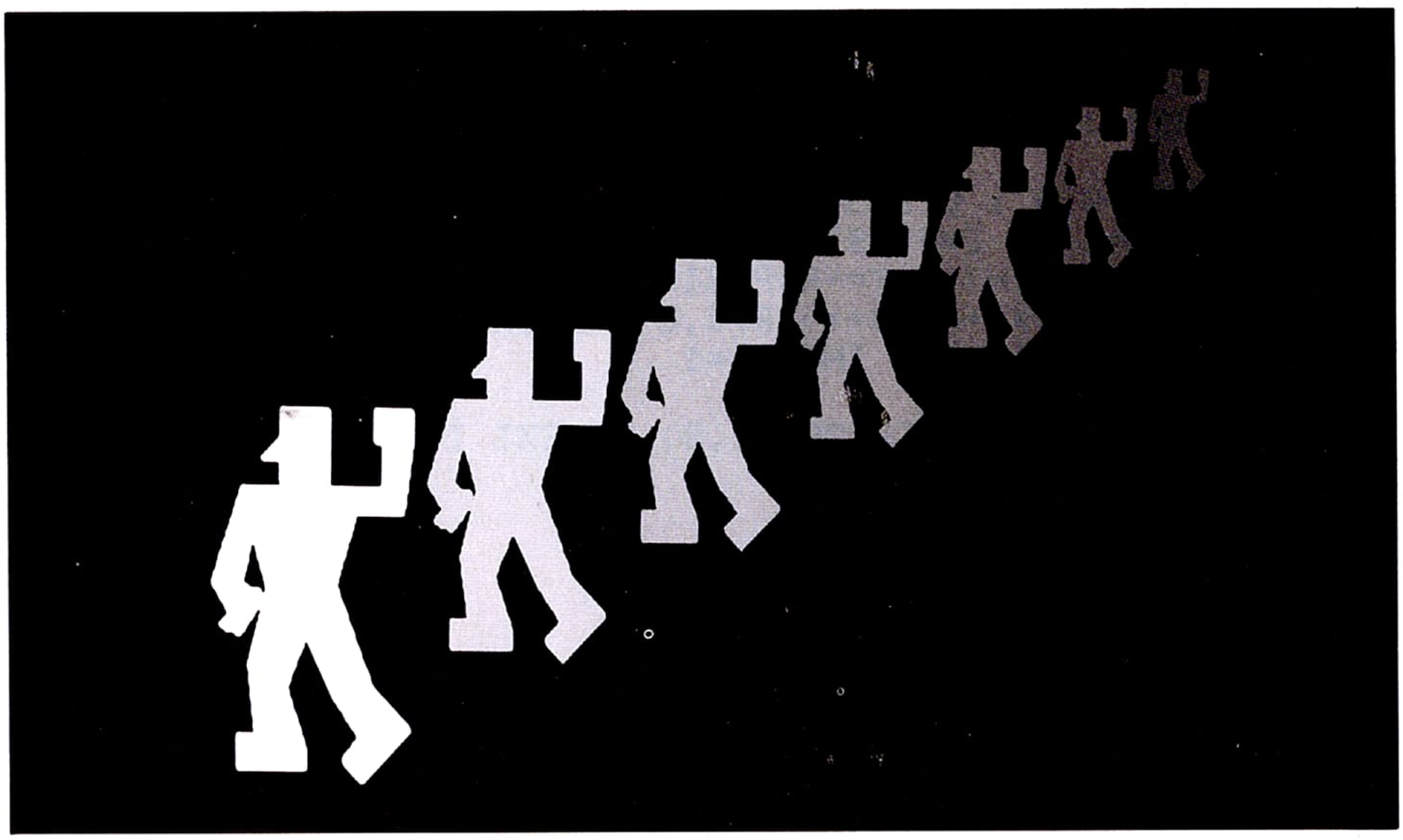
図形の拡大・縮小

ある図形を描くときに，それをもう少し大きくしたいとか，もっと小さく描けたらいいのにと思うことはよくあることです．そこで，図形の拡大・縮小について考えてみましょう．図形の拡大・縮小は，相対座標を用いれば実現します．例えば，*Figure 1* のように，ある四角形を考えると，この四角形の各辺を2倍するには，各頂点の座標に2を掛ければよいのです．

それでは，*List 4.5* を見てください．

340行以降のDATAは，相対座標を用いて表した図形の座標データです．今までのプログラムでは，このデータをただ単に始点の座標と足して出力していました．しかし，ここでは図形の拡大をするため，読み取ったデータに，倍率の変数Mを掛けてから出力します．この変数を1より大きくすれば図形は拡大され，1より小さくすれば縮小されます．280行の変数Mが，その倍率です．

なお，2章でも説明したように，WINDOW命令とVIEW命令を使っても，図形の拡大・縮小は可能です．



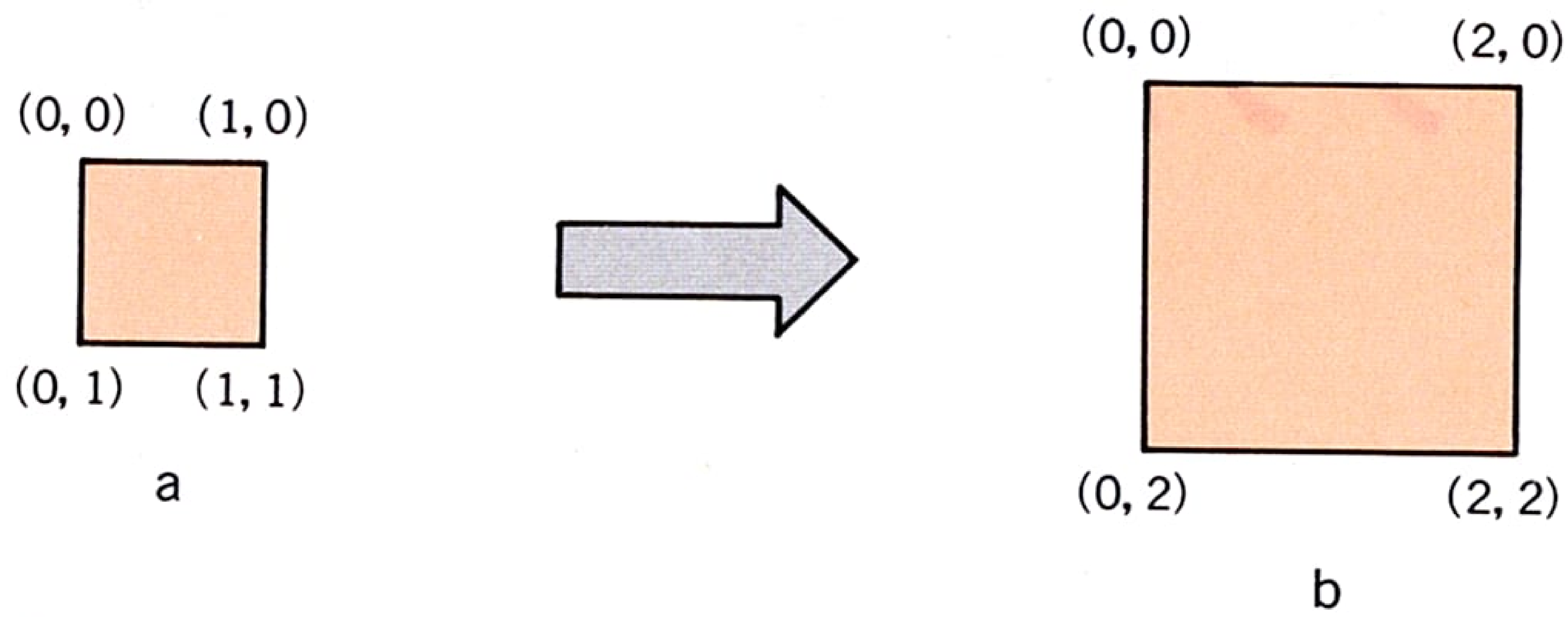


Figure 1

List 4.5

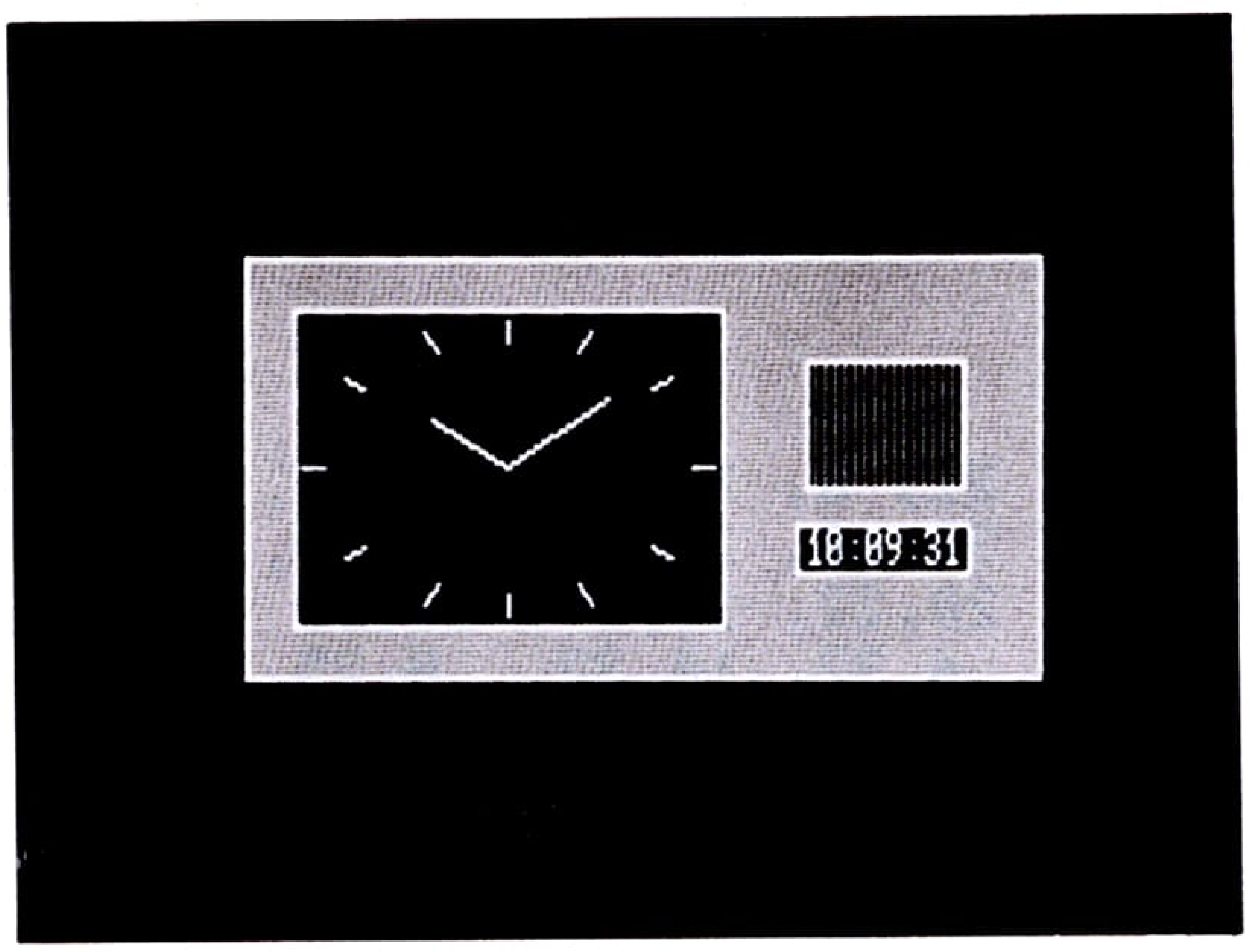
```
100 ' List 4.5
110 '
120 X=620 : Y=0 : C=1
130 DIM X(48),Y(48)
140 FOR I=1 TO 48
150   READ X(I),Y(I)
160 NEXT I
170 '
180 SCREEN 0,0 : CLS 3
190 FOR M=.9 TO 2.7 STEP .3
200   X=X-M*45 : Y=Y+M*8 : GOSUB *MAN
210   FOR T=1 TO 100 : NEXT T
220 NEXT M
230 END
240 '
250 *MAN
260 POINT (X+X(1)*M,Y+Y(1)*M)
270 FOR I=2 TO 48
280   LINE -(X+X(I)*M,Y+Y(I)*M),C : BEEP 1 : BEEP 0
290 NEXT I
300 PAINT (X+24*M,Y+20*M),C
310 C=C+1 : IF C=8 THEN C=1
320 RETURN
330 '
340 DATA 14, 0, 24, 0, 24, 7, 22, 7, 22, 8, 36, 8, 36, 3, 34, 3
350 DATA 34, 0, 42, 0, 42, 3, 40,10, 30,10, 26,17, 34,27, 42,31
360 DATA 32,36, 28,34, 32,32, 26,28, 20,22, 16,27, 20,33
370 DATA 20,35, 20,33, 20,35,  6,35,  6,32, 10,32,  8,25, 12,18
380 DATA  8,12,  6,17,  8,18, 10,18, 10,19,  6,22,  2,21,  4,20
390 DATA  0,17,  6, 8, 16, 8, 16, 7, 14, 7, 14, 5, 10, 5, 14, 3
400 DATA 14, 0
```

人を拡大するルーチン（行 260〜320）

アナログ時計を作る

PC-8801 mk IIの内部には，時計があります．この時計の時間は，TIME$で読むことができます．この機能を利用して，自分だけの時計を作ってみましょう．

このプログラムでは，140〜170 行で，時計の枠を描いています．さらに 180，190 行では，目覚しのスピーカーの部分とその周囲を中間色ペイントで塗っています．220〜280 行は，時計の目盛りを書き込んでいる部分です．DEG という変数が 12 時の位置からの中心角を表しており，この値を時計まわりに 30 度ずつ増やしていくことで目盛りの座標を求めています．N_{88}-BASICでは三角関数で扱う数の単位はラジアンなので「度→ラジアン」の変換を行うために 230 行で変数 DEG に定数 RPD を掛けています（*Figure 1* 参照）．



300 行以降が，針を描き換えていく部分です．ここでは，まず変数 H に時，M に分の値を入れるために，MID$により，TIME$からそれぞれの値を求めます．この値をもとに，時刻に対応する長針，短針の 12 時からの中心の角度を，340 行で求めています．しかし，1 回前に描いた長針の角度 TMO が，新しい長針の TMN と等しい場合には，再び TIME$を読むために 310 行へ戻ります．TMO≠TMN の場合，まず 370 行で，前の長針，短針を消してから，380 行で，次の長針，短針を描きます．なお 335 行に，

```
IF H=7 AND M=0 THEN BEEP
```

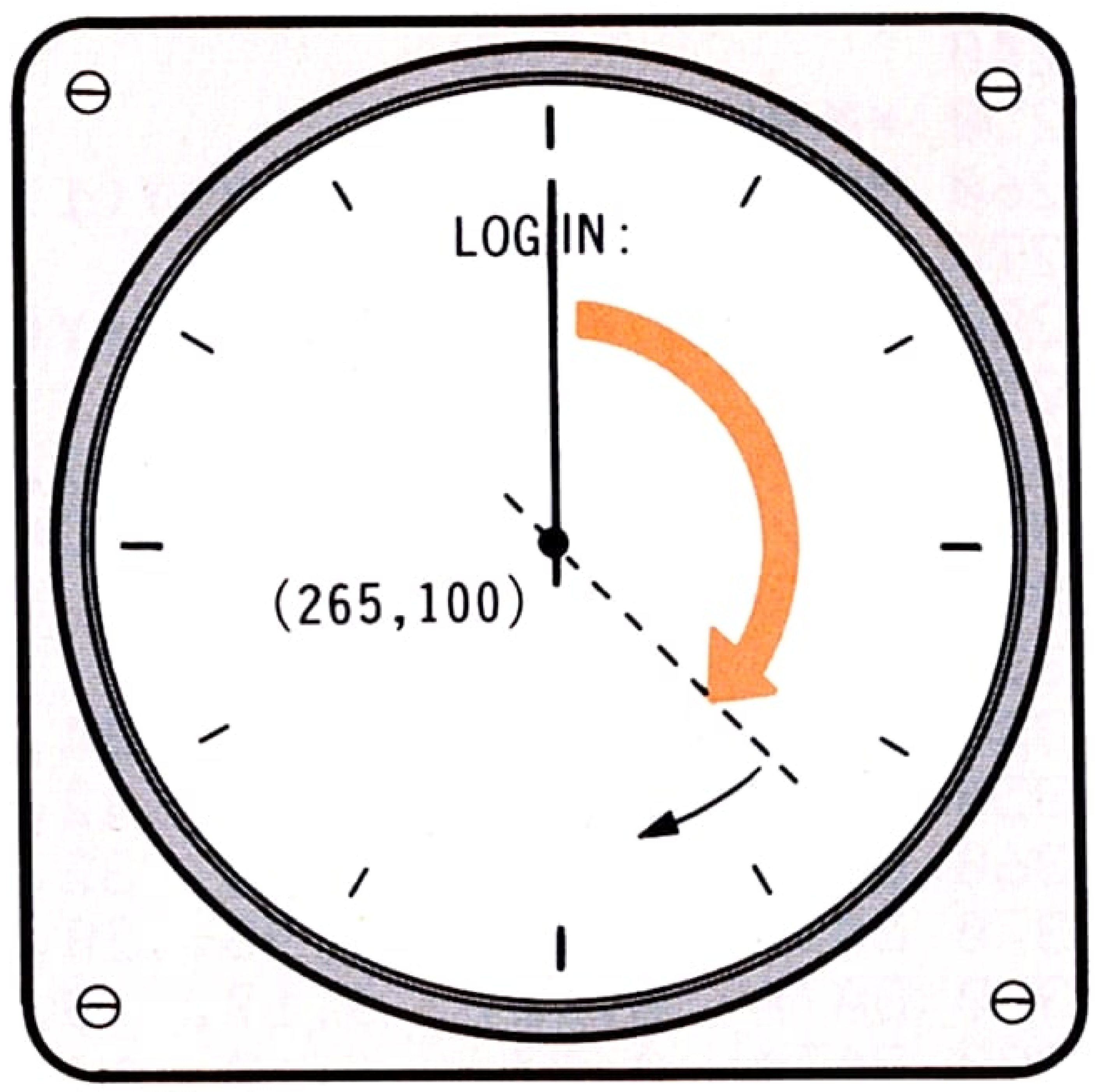


Figure 1

などと加えると，7時にBEEP音がして，目覚し時計の代わりにもなります．いろいろな時計を考えてみてください．プログラムは，何かキーを押すと終了します．

List 4.6

```
100 ' List 4.6
110 '
120 WIDTH 80,25 : CONSOLE ,,0,0 : SCREEN 0,0 : CLS 3
130 COLOR 0
140 LINE (160,60)-(480,140),7,B
150 LINE (180,70)-(350,130),7,B
160 LINE (384,80)-(448,104),7,B
170 LINE (380,111)-(450,120),7,B
180 PAINT (170,65),CHR$(85)+CHR$(0)+CHR$(255),7
190 PAINT (400,90),CHR$(17)+CHR$(17)+CHR$(0),7
200 '
210 PI=3.14159 : RPD=PI/180 : P=.4495 : XW=82 : YW=62
220 FOR DEG=0 TO 359 STEP 30
230   RAD=DEG*RPD : SN=SIN(RAD) : CN=COS(RAD)
240   R=1/SQR((SN/XW)^2+(CN/YW)^2)
250   X1=265+R*SN : Y1=100-R*P*CN
260   X2=265+(R-9)*SN : Y2=100-(R-9)*P*CN
270   LINE (X1,Y1)-(X2,Y2),7
280 NEXT DEG
290 '
300 RH=50 : RM=36
310 *LOOP
320 IF INKEY$<>"" THEN END
330 T$=TIME$ : H=VAL(MID$(T$,1,2)) MOD 12 : M=VAL(MID$(T$,4,2))
340 THN=(30*H+.5*M)*RPD : TMN=6*M*RPD
350 LOCATE 48,14 : PRINT TIME$
360 IF TMO=TMN GOTO *LOOP
370 BEEP 1 : C=0 : A=THO : B=TMO : GOSUB *HAND
380 BEEP 0 : C=7 : A=THN : B=TMN : GOSUB *HAND
390 THO=THN : TMO=TMN : GOTO *LOOP
400 '
410 *HAND : COLOR=(3,6)
420 LINE (265,100)-(265+RM*SIN(A),100-RM*P*COS(A)),C
430 LINE (265,100)-(265+RH*SIN(B),100-RH*P*COS(B)),C
440 COLOR=(3,3) : RETURN
```

140〜170：時計の枠を描く

180〜190：中間色でスピーカーとその周囲を描く

220〜280：時計の目盛りを描くルーチン

330〜390：針を描く位置を算出する

420〜430：針を描く

迷路を作る

誰でも，紙の上に書かれた迷路を鉛筆の先でたどっていく，という経験が，1度や2度はあると思います．

この迷路をコンピュータに作らせるにはどうすればよいのでしょうか．

迷路を作るには，道をのばしていく方法と，壁をのばしていく方法がありますが，ここでは壁をのばしていく方法を使います．*Figure 1* のように入口と出口によって，外壁は2つに分けられます．この外壁（壁a，b)に接する壁を次々に作っていくのですが，既に存在している壁と壁をつなぐような壁を作ってはいけません．抜け道がふさがってしまうことがあるからです．この

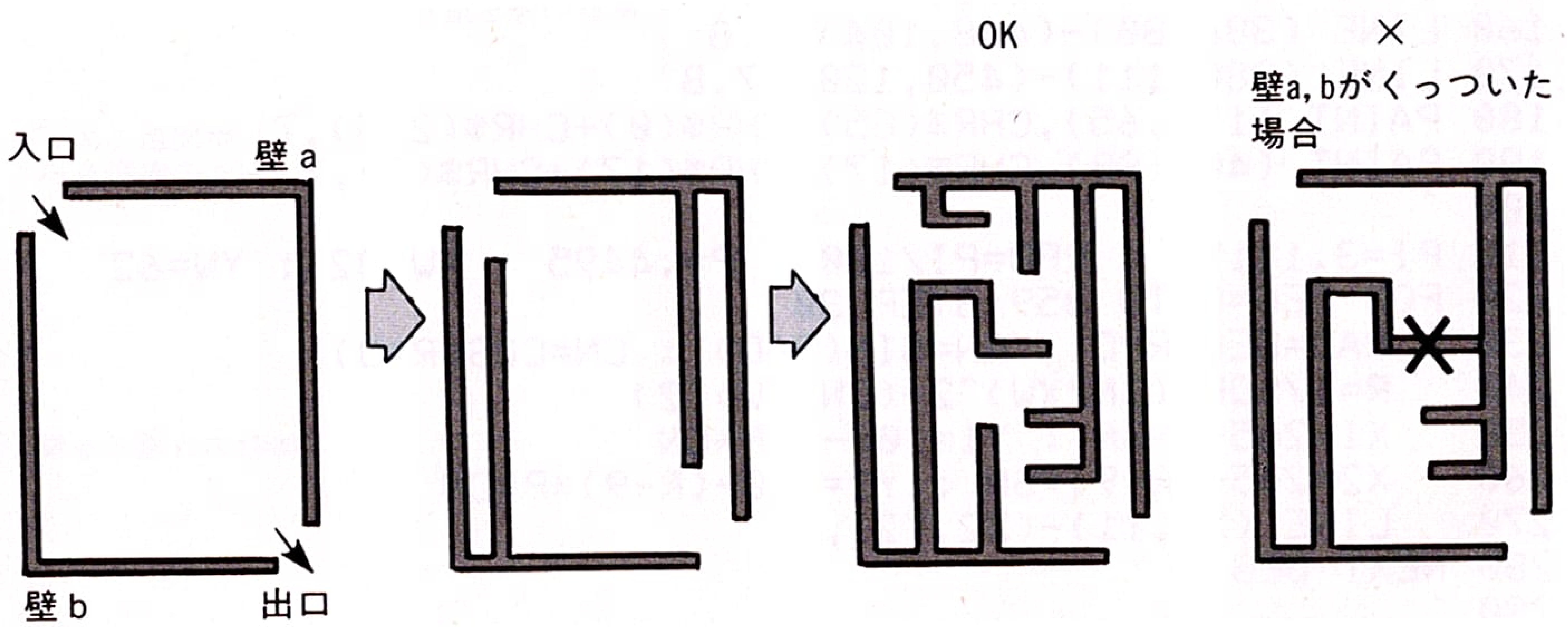
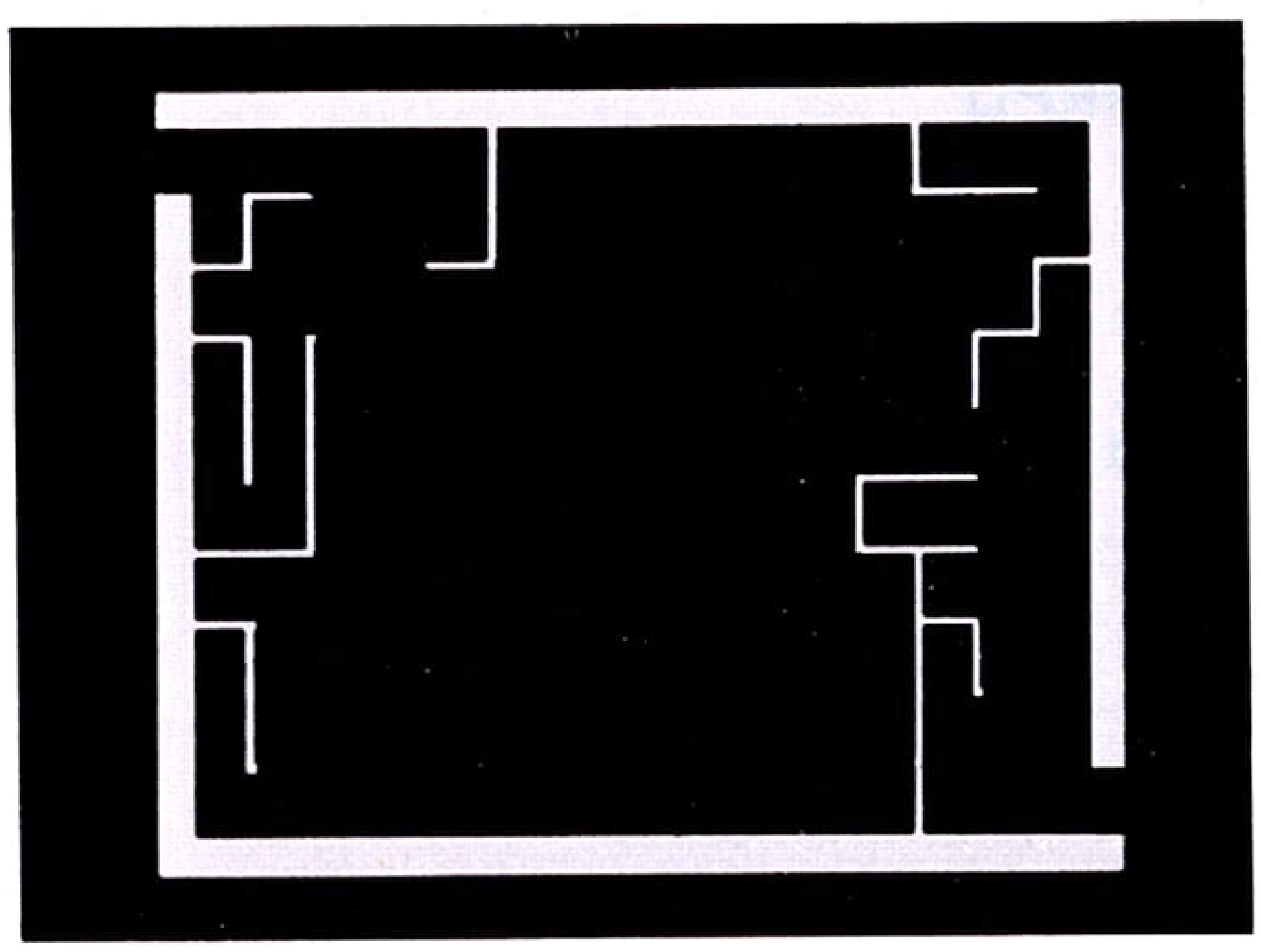
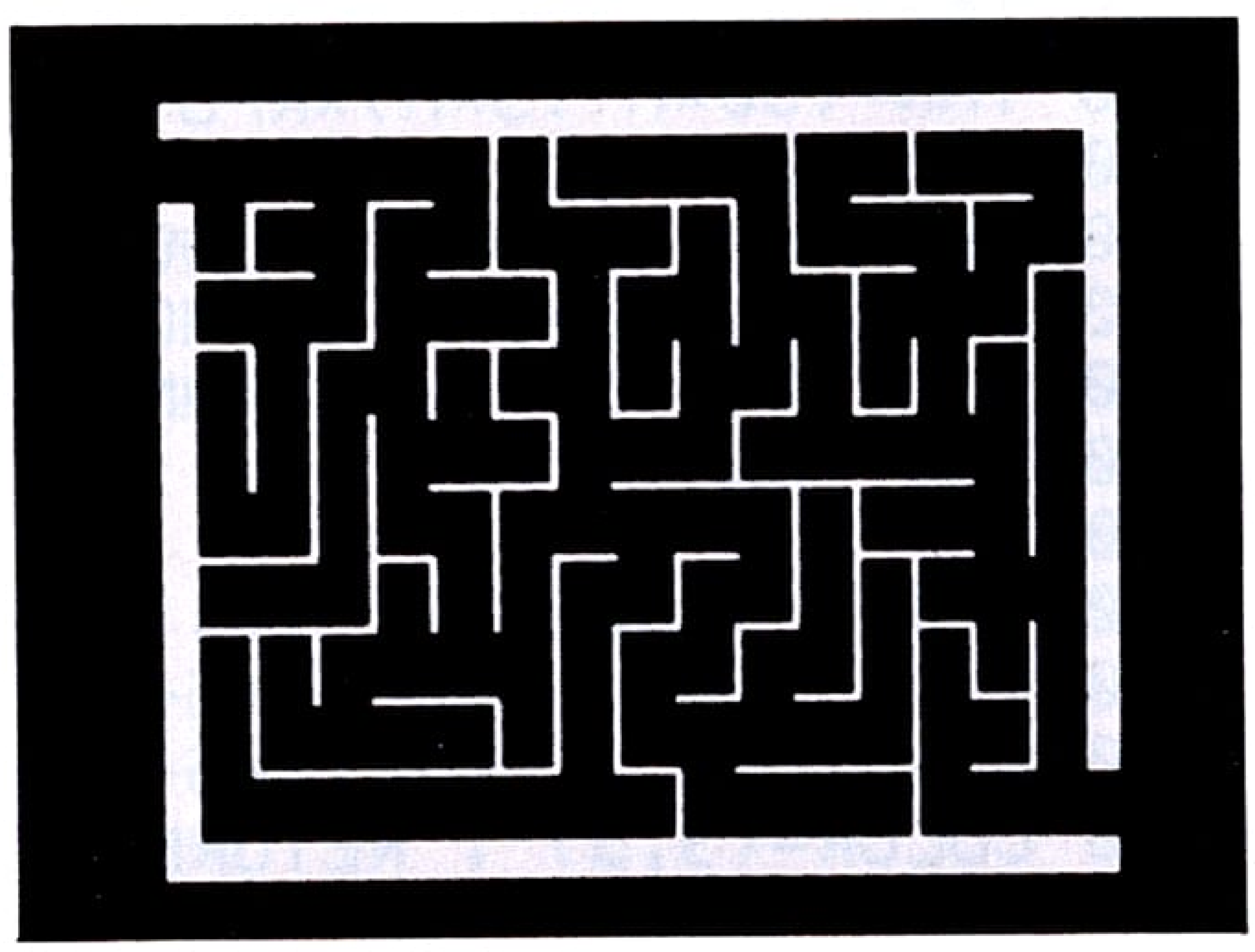


Figure 1





原則さえ守って壁を作っていけば，必ず抜け道のある迷路ができあがります．

このプログラムでは，230〜250 行で外壁を描き，260，270 行で入口と出口を消しています．300〜320 行は，2 つの壁のどちらか一方に接する点を探しています．その点が決まると，340 行で発生させた乱数により，どちら向きの壁を作るか決定し，350〜380 行でその壁が他の壁と接しないかどうか判定し，390 行で壁を描きます．

この方法は，大きい迷路を作ろうとすると非常に時間がかかるのが欠点です．他にも迷路の描き方はありますから，研究してみてください．

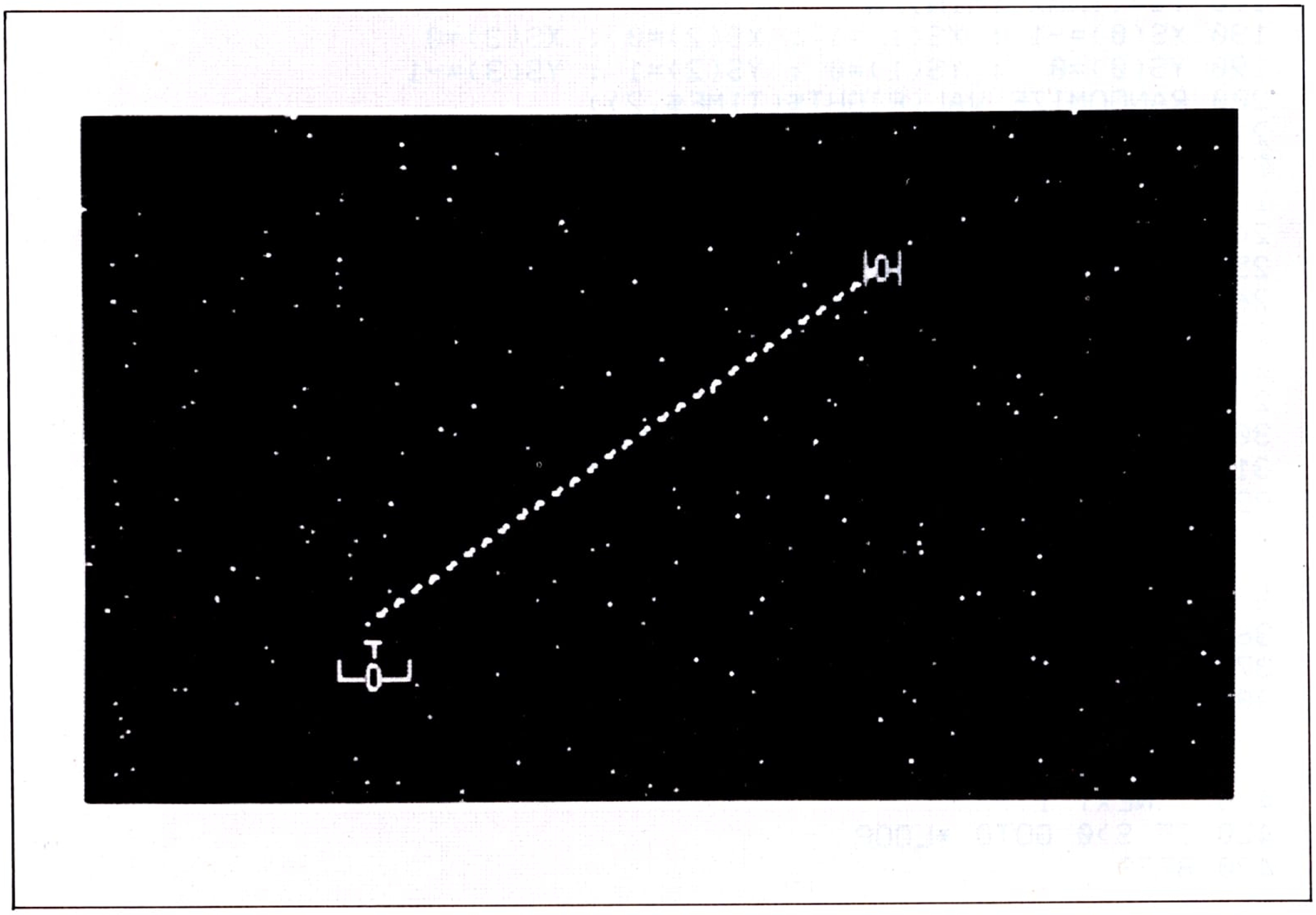
List 4.7

```
100 ' List 4.7
110 '
120 XMIN=160 : YMIN=30
130 XMAX=480 : YMAX=150
140 M=15 : N=10
150 S=(M-1)*(N-1)
160 XW=(XMAX-XMIN)/M
170 YW=(YMAX-YMIN)/N
180 XS(0)=-1 : XS(1)=1 : XS(2)=0 : XS(3)=0
190 YS(0)=0  : YS(1)=0 : YS(2)=1 : YS(3)=-1
200 RANDOMIZE VAL(RIGHT$(TIME$,2))
210 '
220 SCREEN 0,0 : CLS 3
230 LINE  (XMIN-10,YMIN-5)-(XMAX+10,YMAX+5),7,B             ┐
240 LINE  (XMIN,YMIN)-(XMAX,YMAX),7,B                       ├ 外壁を描く
250 PAINT (XMIN-1,YMIN-1),CHR$(85)+CHR$(85)+CHR$(255),7     ┘
260 LINE  (XMIN-10,YMIN+1)-(XMIN,YMIN+YW-1),0,BF  ┐
270 LINE  (XMAX,YMAX-1)-(XMAX+10,YMAX-YW+1),0,BF  ┘ 出入口を描く
280 '
290 *LOOP
300   X=INT(RND*(M+1)) : Y=INT(RND*(N+1))
310   XO=XMIN+X*XW     : YO=YMIN+Y*YW
320   IF POINT(XO,YO)=0 GOTO *LOOP
330   FOR I=1 TO 8
340     R=INT(RND*4)……壁をのばす方向を決定
350     XN=XO+XS(R)*XW : YN=YO+YS(R)*YW      ┐
360     IF XN<=XMIN OR XN=>XMAX GOTO 410     │ あらたに作る壁が他の壁に接するか
370     IF YN<=YMIN OR YN=>YMAX GOTO 410     │ どうかを判定する
380     IF POINT(XN,YN)<>0 GOTO 410          ┘
390     LINE (XO,YO)-(XN,YN),7
400     XO=XN : YO=YN : S=S-1
410   NEXT I
420 IF S>0 GOTO *LOOP
430 BEEP
```

グラフィック・キャラクタで宇宙船を！

マニュアルのASCIIコード表や，キーボードの配列表を見ると，なにやら●，○，♥，■，などいろんな形があることに気づくでしょう．これがグラフィック・キャラクタです．ここではこのグラフィック・キャラクタで，宇宙船を描いて，動かしてみます．キーボードからこれらのキャラクタを入力するには，GRPHキーを押しながら，目的のキーを押します．

ただし，ASCIIコード表の始めの方にある↑↓→←などの記号は例外です．これは，カーソルをおのおのの方向へ移動するという役目を持ったコードで，キーボードの右上のカーソルキーに対応しています．このコードをCHR$関数でキャラクタに変換し，PRINT命令で表示しても，矢印は



画面に出ません.

プログラムを見てください. 130行のA$とB$で2つの図形を作っています. CHR$（n）が何の文字に対応しているかは, ASCIIコード表を見てください. この文字列は, LOCATE命令でその始点を指定することができるので, 画面上の好きな位置にその図形を描くことができます.

このプログラムでは, 図形の動きに変化を与えるために, 240～320行のFOR～NEXTループ内で三角関数を使い, 2つの図形の始点, X1, X2を決めています. 300行でキー入力があれば, 350行以降のサブルーチンで, ビームを描かせています.

このプログラムは, 少し手を加えると簡単なゲームをなりそうですね. 図形の定義などを変えて遊んでみてください. なお, ビーム発射はスペースキー, 終了するには［ESC］キーを押します.

List 4.8

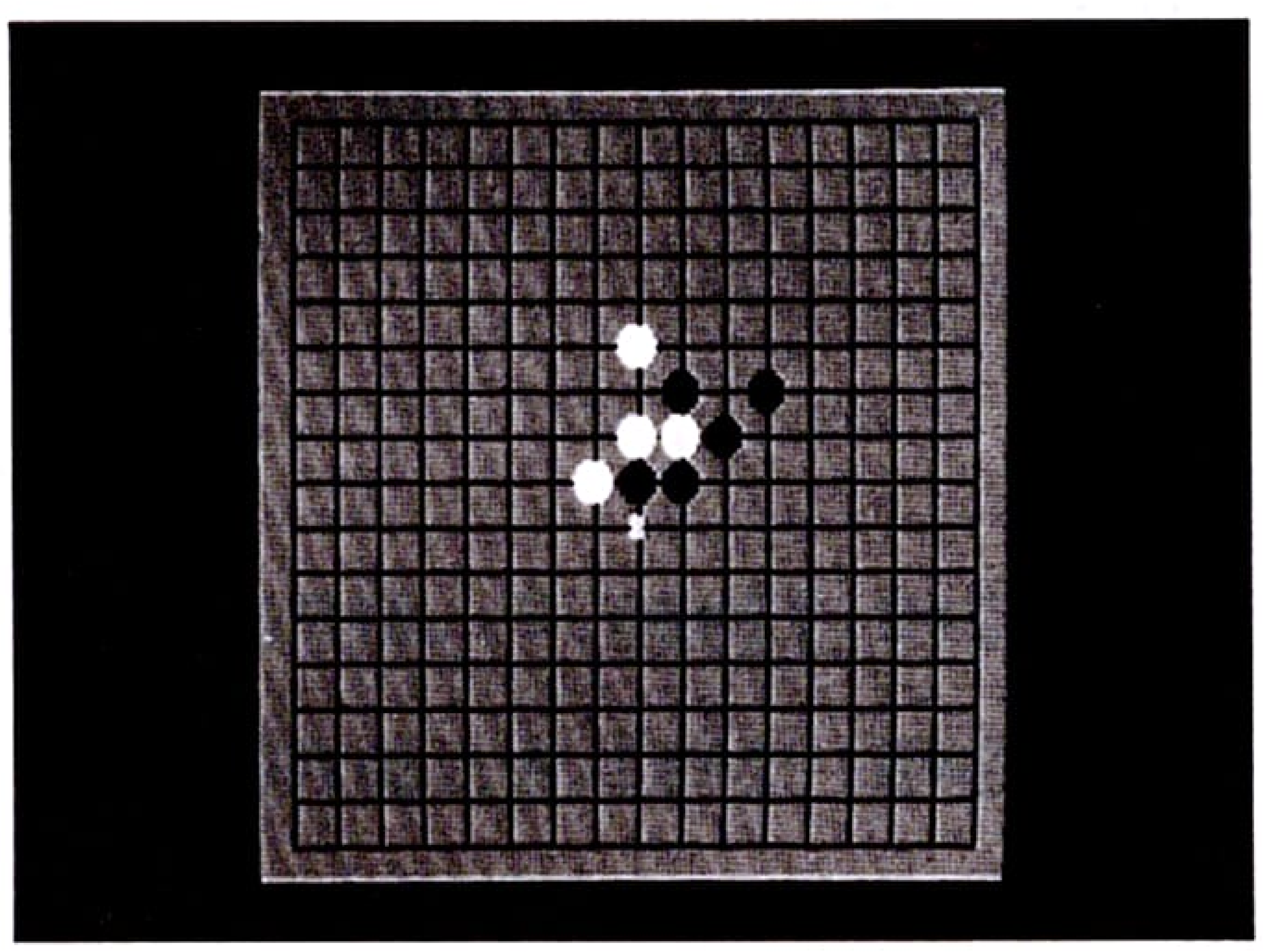
```
100 ' List 4.8
110 '                 ┌─定数の設定
120 PI=3.14159 : XM=40 : R=20 : Y1=7 : Y2=17
130 A$="├O┤": B$="┬"+CHR$(31)+STRING$(3,29)+"└─O─┘"─┐
140 '                                                キャラクタを作る
150 WIDTH 80,25 : CONSOLE ,,0,1 : SCREEN 0,0 : CLS 3
160 COLOR 7  : OUT 230,0
170 '
180 FOR I=1 TO 400
190   PSET ((RND*540)+50,INT(RND*150)+25),INT(RND*7)+1 } 星を描く
200 NEXT I
210 LINE (50,25)-(590,175),7,B
220 PAINT (0,0),CHR$(255)+CHR$(0)+CHR$(85),7 } 外枠を中間色で描く
230 '
240 FOR I=0 TO 2*PI STEP PI/16
250   X1=XM+INT(R*SIN(I))
260   X2=XM+INT(R*COS(I))
270   COLOR 3 : LOCATE X1,Y1 : PRINT A$
280   COLOR 5 : LOCATE X2,Y2 : PRINT B$
290   IF INP(9)=127 THEN COLOR 7 : OUT 230,3 : END } キャラクタを動かす
300   IF INP(9)=191 THEN GOSUB *BEEM
310   CLS
320 NEXT I
330 GOTO 240
340 '
350 *BEEM
360 BEEP 1 : LINE (X1*8+8,Y1*8+4)-(X2*8,Y2*8),7,,&HF0F0
370 FOR T=1 TO 50 : NEXT T                                } 発射されたビームを描く
380 BEEP 0 : LINE (X1*8+8,Y1*8+4)-(X2*8,Y2*8),0,,&HF0F0
390 RETURN
```

五目並べはいかが？

五目並べは，碁盤に黒石と白石を交互に置くことを繰り返して，先に5つ同じ色の石を並べた方が勝ちです．このゲームのプログラムで問題なのは，石を置く場所を間違えたときに，その石を取り除く作業です．

石を置くだけであれば，位置を決めてCIRCLE命令で円を描き，その内部をPAINT命令を使って黒か白に塗ればよいでしょう．しかし，置いた石を取り除くとなると，下地の碁盤のマス目が消されてしまわないようにする処理が必要になります．

そこで，あらかじめ石の形をGET命令で配列に蓄えておき，それを碁盤にPUT命令のXOR機能で出力します．置いた石を取り除きたいときは，再びXOR機能で同じ色の石を，取り去りたい石の上に重ねて描けば，下地のマス目が石を置く以前と同様に現われてきます．これがXOR機能の特徴です．なお，PUT命令を使うときに，



PSET，PRESETなどを指定しないと，XOR機能が指定されたことになります．

しかし，XOR機能で石を出力しただけでは，石の色は，黒と白になりません．ここで*List 4.9*のプログラムを実行させて，いくつか石を置いてから，終了させてみてください（プログラムは[ESC]キーで終了します）．終了させると同時に石の色が変わりますよね．XOR機能だけで石を出力すると，こういう色のままで，ゲームをすることになります．そこで，1章で説明したようにPC-8801 mk IIのパレット機能を利用して，見た目に石が黒か白になるように，パレットコードを決めてしまいます．いま下地の碁盤は，黄色と黒で描かれています．ここに青色の石をXOR機能で出力すると，黄色の部分は白に，黒の部分は青になってしまいます．そこで，パレットコードの白と青を黒に対応させてしまえば，黒の石を置いたことになります．白石も同じ考え方で，赤で出力し緑と赤のパレットコードを白色に対応させて，白い石が現れるようにします．なかなか理解するには難しいところだと思いますが，リストをよぉくにらんで研究してみてください．

最後に，キー操作について述べておきます．十字のマークはカーソルキーで動かします．[B]キーで黒石，[W]キーで白石が，マス目に置かれます．あらかじめ石の置いて

ある場所には他の石は置けません．このときはスペースキーを押してその石を取り除いてから，好みの色の石を置いてください．勝ち負けの判定はしませんので，念のため．

List 4.9

```
100 ' List 4.9
110 '
120 XMIN=170 : YMIN=20  : XMAX=490 : YMAX=180 : N=17
130 XW=(XMAX-XMIN)/(N-1) : YW=(YMAX-YMIN)/(N-1)
140 M=(((XW+8)¥8)*(YW+1)*3+4)¥4+1
150 DIM D(N,N),CR(M),B(M),W(M)
160 KY$=CHR$(28)+CHR$(29)+CHR$(30)+CHR$(31)+"BbWw  "+CHR$(27)
170 XS(0)=0 : XS(1)=1 : XS(2)=-1 : XS(3)=0 : XS(4)=0
180 YS(0)=0 : YS(1)=0 : YS(2)=0  : YS(3)=-1: YS(4)=1
190 '
200 SCREEN 0,0 : CLS 3 : LOCATE ,,0
210 CIRCLE (XW/2,YW/2),XW/2-2,1         : PAINT(XW/2,YW/2),1
220 GET (0,0)-(XW,YW),B
230 CIRCLE (XW*1.5,YW*1.5),XW/2-2,2 : PAINT(XW*1.5,YW*1.5),2
240 GET (XW,YW)-(XW*2,YW*2),W
250 LINE (0,0)-(4,4),3,BF : GET (0,0)-(4,4),CR
260 COLOR=(7,0) : COLOR=(1,0) : COLOR=(4,7) : COLOR=(2,7)
270 CLS 3
280 LINE (XMIN-15,YMIN-7)-(XMAX+15,YMAX+7),6,B
290 PAINT (XMIN,YMIN),CHR$(0)+CHR$(85)+CHR$(85),6
300 FOR I=0 TO N-1
310   LINE (I*XW+XMIN,YMIN)-(I*XW+XMIN+1,YMAX),0,BF
320   LINE (XMIN,I*YW+YMIN)-(XMAX,I*YW+YMIN),0
330 NEXT I
340 XN=(N+1)/2 : YN=XN : GOTO 410
350 '
360 *LOOP : I=INSTR(KY$,INPUT$(1)) : BEEP 1 : BEEP 0
370   IF I>4 THEN ON I-4 GOTO *B,*B,*W,*W,*E,*F
380   XN=XO+XS(I) : YN=YO+YS(I)
390   IF XN*YN*(N+1-XN)*(N+1-YN)<=0 THEN XN=XO:YN=YO:GOTO *LOOP
400   GX=XMIN+(XO-1)*XW : GY=YMIN+(YO-1)*YW : GOSUB *C
410   GX=XMIN+(XN-1)*XW : GY=YMIN+(YN-1)*YW : GOSUB *C
420   XO=XN : YO=YN : GOTO *LOOP
430 '
440 *C : PUT (GX-2,GY-2),CR : RETURN
450 *B : IF D(XN,YN)<>0 THEN *LOOP ELSE D(XN,YN)=1  : GOSUB *C
460      PUT (GX-XW/2,GY-YW/2),B : GOSUB *C : GOTO *LOOP
470 *W : IF D(XN,YN)<>0 THEN *LOOP ELSE D(XN,YN)=-1 : GOSUB *C
480      PUT (GX-XW/2,GY-YW/2),W : GOSUB *C : GOTO *LOOP
490 *E : IF D(XN,YN)=0  THEN *LOOP ELSE GOSUB *C
500      IF D(XN,YN)=1  THEN PUT (GX-XW/2,GY-YW/2),B
510      IF D(XN,YN)=-1 THEN PUT (GX-XW/2,GY-YW/2),W
520      GOSUB *C : D(XN,YN)=0 : GOTO *LOOP
530 *F : FOR I=1 TO 7 : COLOR=(I,I) : NEXT I : LOCATE ,,1 : END
```

碁石を配列に読み込む（210〜240）
碁板を描く（280〜290）
マス目を描く（310〜330）
キーからの入力によって碁石を置く（360〜420）
パレットコードを初期状態に戻す（440〜530）

Night Flight

複雑な図形を，素早く，簡単に動かすには，あらかじめ描いておいた図形をGET命令で蓄えておき，以後PUT命令で出力する方法がよいでしょう．ここではその例として，夜空に飛行機を飛ばすプログラムを作ってみました．

List 4.10 を見てください．180〜220行で飛行機を描き，240行のGET命令で配列変数PLANEに読み込んでいます．このデータは，410行で飛行機を出力するときに利用されます．

飛行機を移動させるとき，その背景を消してしまわないように，以下の手順を踏みます．

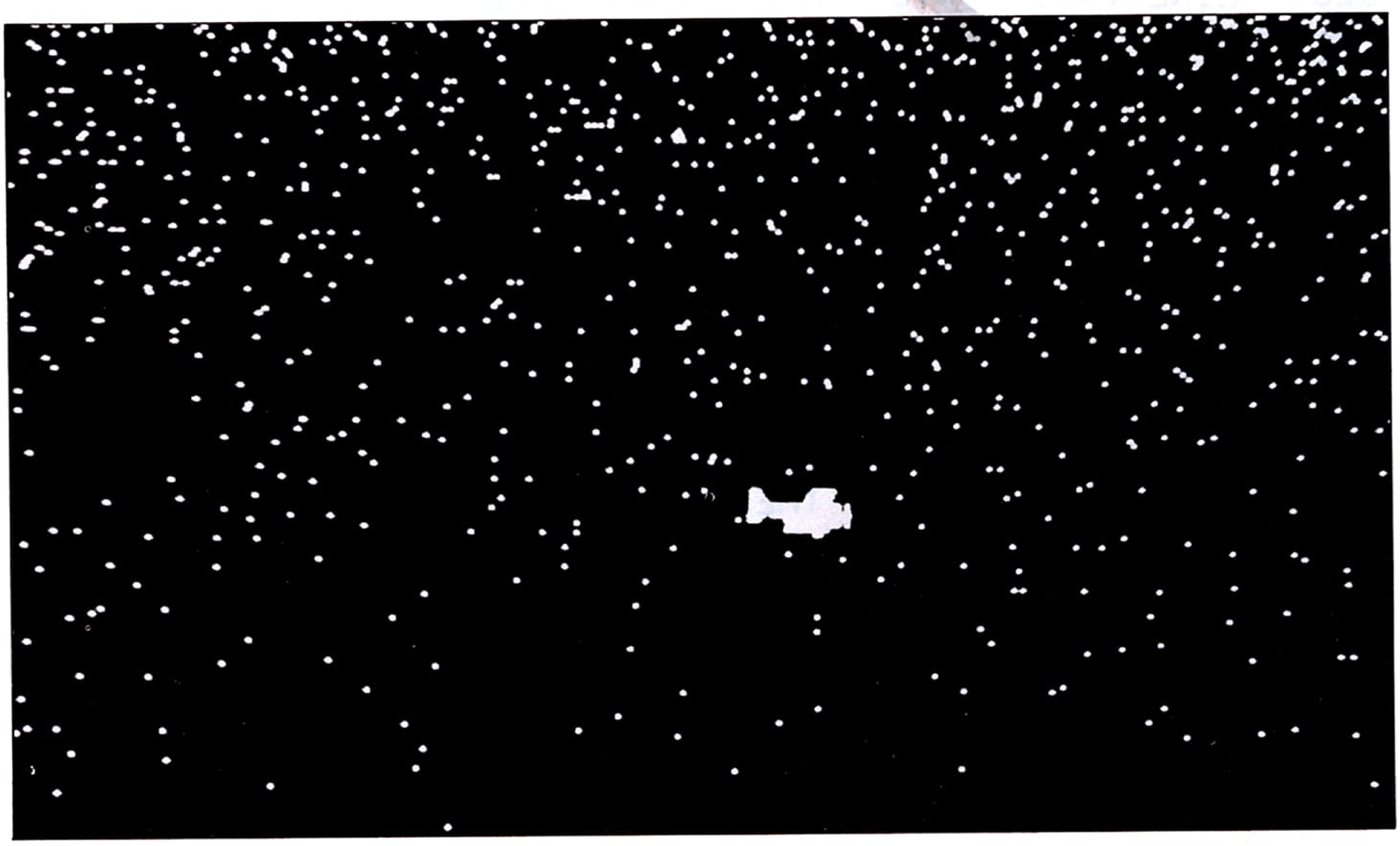
(1) 表示された飛行機の上にBACKのデータを出力して飛行機を消すため，背景を描く(390行)

(2) 次に飛行機が描かれる部分の背景を配列変数BACKにGET命令で蓄えておく（400行）

(3) (2)で蓄えられた座標に飛行機を描く（410行）

(4) (1)へ戻る

この一連の処理の途中でキー入力を行い，飛行機を上下に動かしています．カーソルキー[↑]で上へ，[↓]で下へと，飛行機が移動



します．終了するときは，[ESC]キーを押してください．

この程度のプログラムなら，ちょっと手を加えるだけでゲームになりそうですね．

List 4.10

```
100 ' List 4.10
110 '
120 XMIN=0    : YMIN=0
130 XMAX=639 : YMAX=199
140 XS=20     : YS=10
150 DIM PLANE(47),BACK(47)
160 '
170 SCREEN 0,0 : CLS 3
180 READ A,B,C,D : LINE(A,B)-(C,D),7
190 READ A,B : IF A=-2 GOTO 220 ELSE IF A=-1 GOTO 180
200   LINE -(A,B),7
210 GOTO 190
220 FOR I=1 TO 6 : READ X,Y,C : PAINT (X,Y),C,7 : NEXT I
230 '
240 GET  (0,0)-(44,9),PLANE
250 LINE (0,0)-(44,9),0,BF
260 '
270 FOR I=1 TO 1000
280   PSET(INT(RND*640),INT(RND*RND*200)),7
290 NEXT I
300 '
310 X0=XMIN : Y0=(YMAX+YMIN)/2
320 GET (X0,Y0)-(X0+44,Y0+9),BACK
330 YN=Y0
340 *LOOP
350   XN=X0+XS : IF XN>XMAX-45 THEN XN=XMIN
360   IF INP(8) =253  AND Y0-YS>YMIN    THEN YN=Y0-YS
370   IF INP(10)=253  AND Y0+YS<YMAX-10 THEN YN=Y0+YS
380   IF INP(9) =127 THEN END
390   PUT (X0,Y0),BACK,PSET
400   GET (XN,YN)-(XN+44,YN+9),BACK
410   PUT (XN,YN),PLANE,PSET
420   FOR T=0 TO 80 : NEXT T
430   X0=XN : Y0=YN : BEEP 1 : BEEP 0
440 GOTO *LOOP
450 '
460 DATA  1,4,  0,6,  3,6,  7,4,  1,4, 1,0, 4,0, 8,3, 1,3,-1,0
470 DATA 28,0, 37,0, 29,8, 16,8, 28,0,-1,0
480 DATA  1,3, 39,3, 39,7, 30,7,-1,0
490 DATA 19,6,  7,4,-1,0 , 43,3, 43,7,-1,0
500 DATA 39,5, 44,5, 39,6,-1,0 , 36.7, 30,9, 28.8.-2,0
510 '
520 DATA  2,5,2, 3,1,4, 30,1,3, 26,6,3, 15,4,6, 35,5,6
```

飛行機を描く（180〜220）

飛行機を配列に取り込む（240〜250）

背景を描く（270〜290）

キー入力によって飛行機を動かすメインルーチン（340〜440）

飛行機のデータ（460〜520）

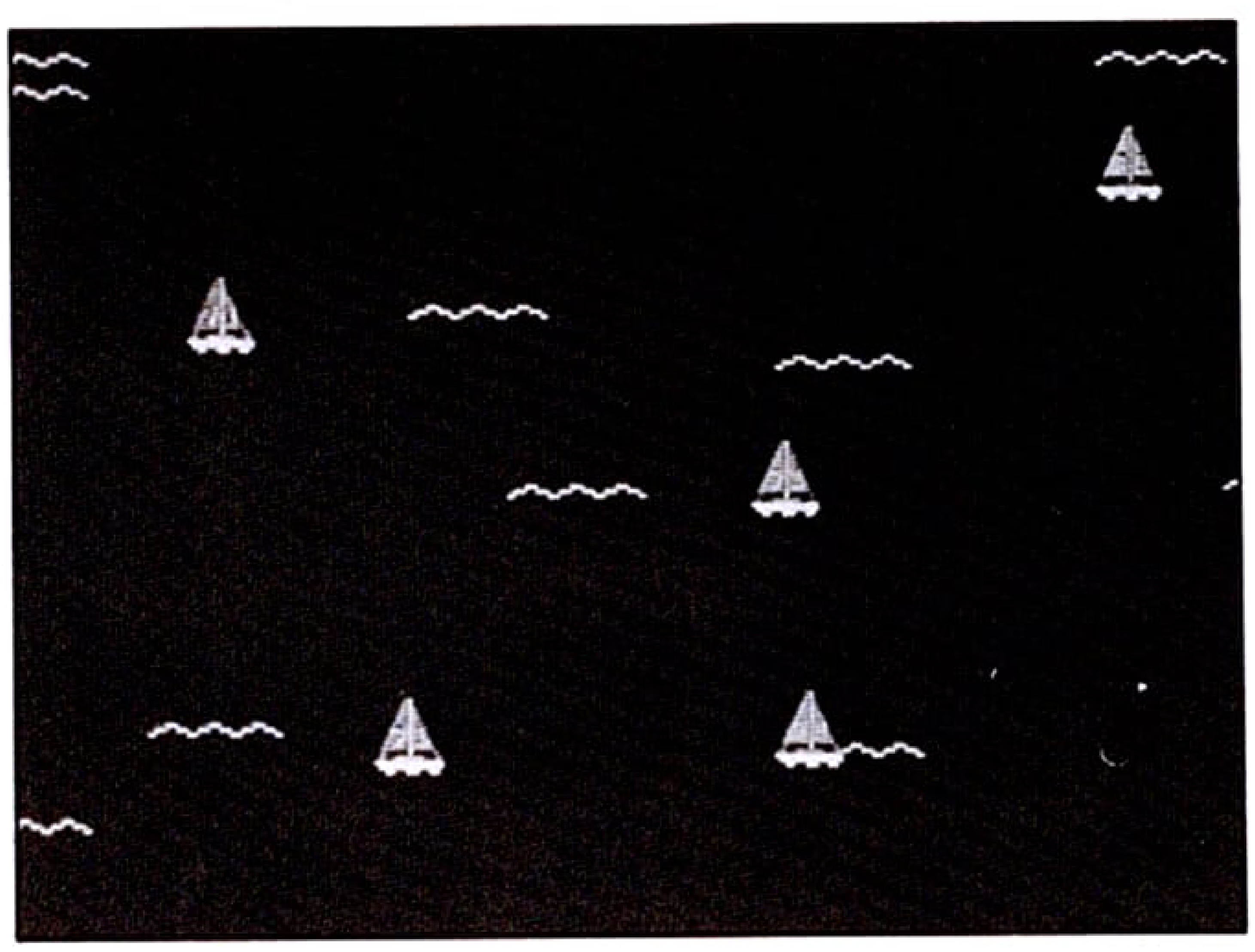
波間にヨットが

次から次へと背景が一定方向に流れていく中で場面が展開していくタイプのゲームを，スクロールゲームと呼びます．狭いディスプレイ画面に閉じ込もることもなく，背景に広大な世界を構築できるスクロールゲームは，ゲームセンターでも常に高い人気を得ています．

ここでは，スクロールの最も基本的なパターンとして，波が寄せる海原に5隻のヨットが浮かんでいるプログラムを作ってみましょう．

まず，波を左から右へ移動していくルーチンについて考えてみます．その手順は，

(1) 波を消す
(2) 波の位置をXS分だけ右へ進める．その座標の値が右端の座標より大きくなったら左端へ戻す
(3) 波を描き直す



(4) (1)の処理へ戻る

波を同時にたくさん動かしたいときは，この処理をひとつひとつの波について行えばよいのですが，仮にその処理をいちいちプログラムに書いたら，それは長大なものになってしまいます．そこで，この場合のように同じ処理を何度も繰り返し行わせるときには，FOR～NEXTループと配列変数を組み合わせると便利です．

実際，240～260行で(1)，(2)，(3)の処理を行い，1つの波を動かします．そして，230～290行のFOR～NEXTループで変数Iの値を1からWNまで変化させながら同様の処理を行い，すべての波の位置を変えていくのです．

ヨットは，そのデータをPUT命令で出力しています．波を移動させたとき，ヨットに重なるとおかしいので270行で1つ波を動かすたびにすべてのヨットを描き直しています．

本格的なスクロールゲームでは，全画面のデータを移動させたりしますが，これをBASICプログラムで組むには，スピードの点で無理があります．これでは満足のいかない方は，マシン語へのアプローチを行ってみるのもよいでしょう．なお，このプログラムは何かキーを押すと終了します．

List 4.11

```
100 ' List 4.11
110 '
120 WN=12 : BN=5 : XS=30
130 DIM X(WN),Y(WN),BX(BN),BY(BN) : GOSUB *GDATA
140 FOR I=1 TO WN
150   X(I)=INT(RND*640) : Y(I)=INT(RND*180)+10
160 NEXT I
170 FOR I=1 TO BN
180   BX(I)=INT(RND*560)+40 : BY(I)=INT(RND*160)+20
190 NEXT I
200 '
210 SCREEN 0,0 : CLS : LINE (0,0)-(639,199),1,BF
220 *LOOP
230 FOR I=1 TO WN
240   C=1 : GOSUB *WAVE
250   X(I)=X(I)+XS : IF X(I)>640 THEN X(I)=-20
260   C=7 : GOSUB *WAVE
270   FOR T=1 TO BN : PUT (BX(T),BY(T)),D%,PSET : NEXT T
280   IF INKEY$<>"" THEN END : LOCATE ,,1
290 NEXT I
300 GOTO *LOOP
310 '
320 *WAVE : POINT (X(I),Y(I))
330 FOR J=1 TO 3
340   LINE -STEP(10,-2),C : LINE -STEP(10,2),C
350 NEXT J : RETURN
360 '
370 *GDATA : DIM D%(99) : RESTORE 400
380 FOR I=0 TO 99 : READ D%(I) : NEXT I : RETURN
390 '
400 DATA 30,16,-1,-769,256,0,1280,0,-1,-769,256,0,3328,0,-1,-769
410 DATA 256,0,7424,64,-1,-769,256,0,15616,96,-1,-769,256,0
420 DATA 32000,112,-1,-769,256,0,-768,120,2047,-825,-1792,56
430 DATA -767,124,-2049,-793,2304,24,-765,126,-12289,-793,12544
440 DATA 24,-761,127,-24577,-829,24832,60,-753,-32641,-1,-769
450 DATA 256,0,-737,-16257,-1,-769,272,8192,256,0,-1,-769,32544
460 DATA 4344,0,0,-1,-769,-1,-769,-1,-769,-1,-769,-193,-769,-193
470 DATA -769,-1,-769,775,-3872,775,-3872,0,0
```

波の位置を決める (150)

ヨットの位置を決める (180)

波を動かす (230–290)

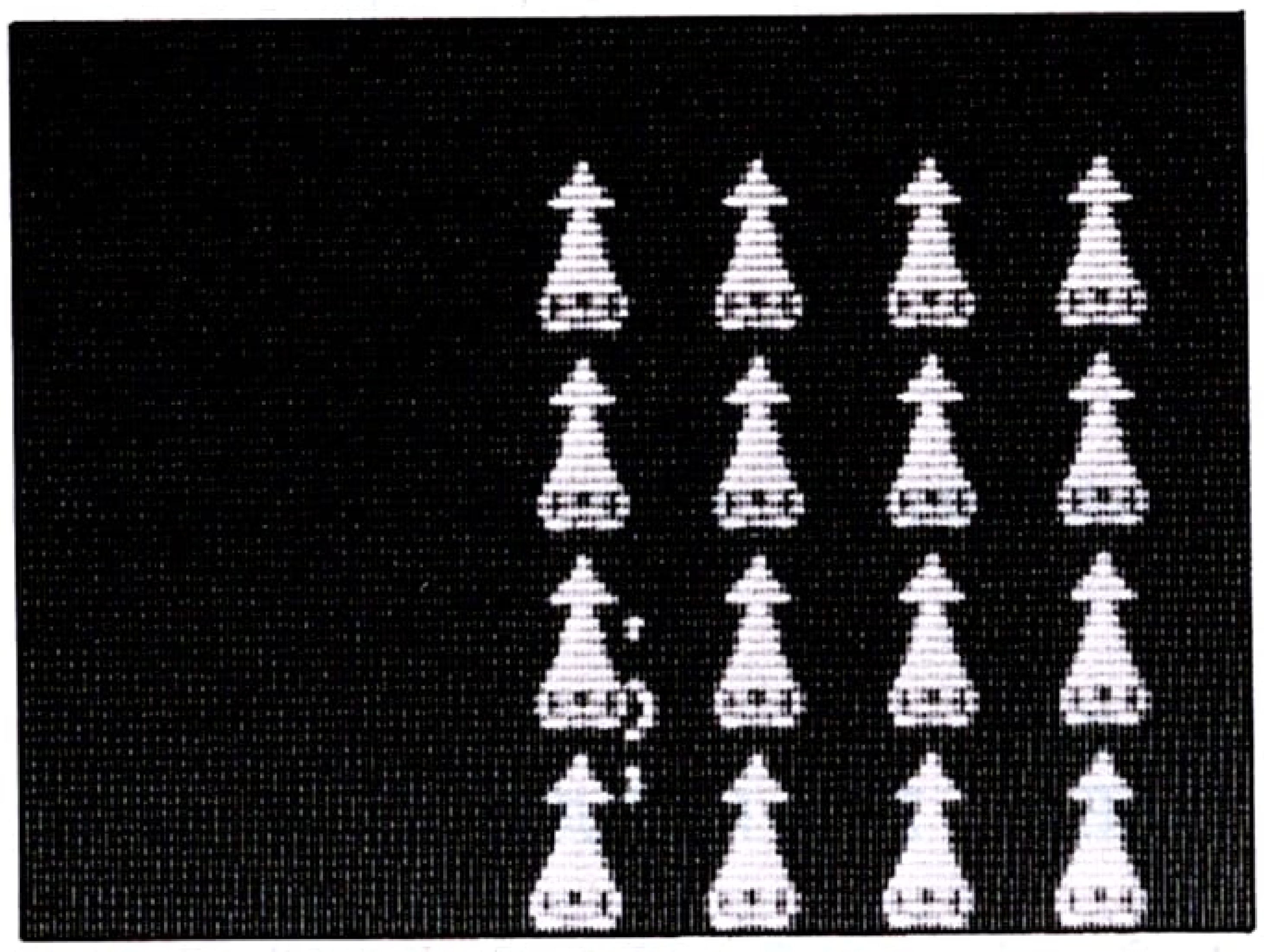
波を描く (320–350)

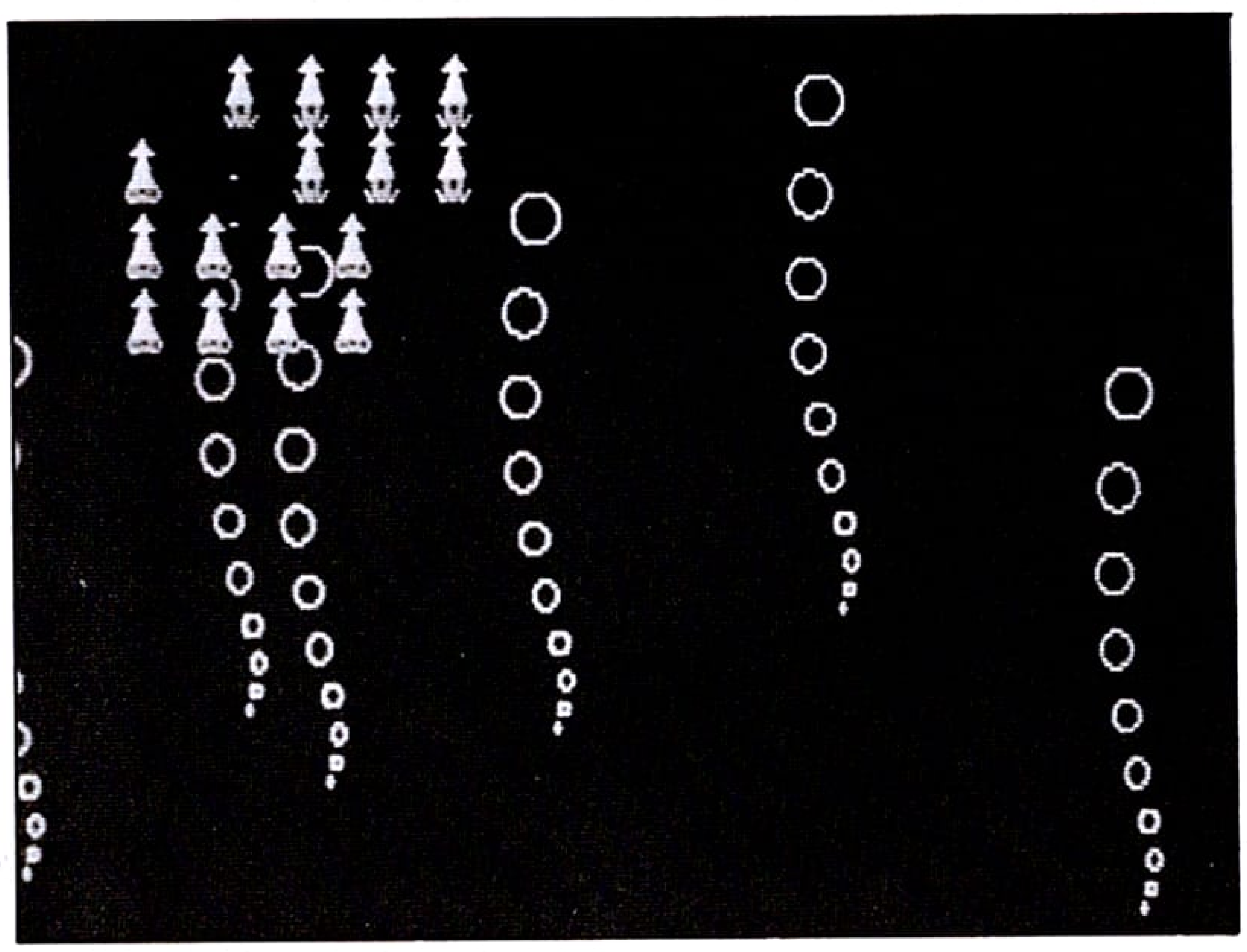
PUTするヨットのデータを配列に読み込む (370–470)

イカの編隊飛行

“波間にヨットが”のプログラムでは，同時に多くの図形を動かすテクニックの基本について述べました．ここではそれを応用して，イカが16匹いっせいに海中を編隊飛行するというロマンチックな？プログラムを作ってみましょう．

まず仮に1匹のイカが画面を左右に動きながら，1段ずつ下へ降りていく場合を考えてみます．1度描いたイカを消してその右隣りにまたイカを描くという処理を繰り返すことでイカは右へ進んでいきます．右端へ達したと判定されると，イカは1段下へ描き直され，今度は左へ進んでいきます．





このイカが，横4匹の編隊になったとしても，1匹1匹の動きにかわりはありません．*List 4.12* を見てください．260～280行がイカの初期設定，320～400行がイカを左右に動かすループ，420～470行がイカたちを1段下へ下げるルーチンになっています．

ただし，1匹のイカではなく，複数個のイカを移動させる場合に，気をつけなければならないことが2つあります．まず，イカの編隊が，右あるいは左に動いていくとき，隣どうしのイカたちが重なり合わないようにする処理が必要です．つまり，編隊が右へ移動していくときは右端のイカから，左へ移動していくときは左端のイカから，それぞれ描き直していくようにするのです（*Figure 1* 参照)．このため右と左の移動では，左右別のイカから動かし始めます．330行と480行でこの処理を行っています．480行で，編隊が1段下がると同時にイカの移動量と向きを表す変数であるXSを−XSとし，さらに330行の変数XBとXEを入れ換えてイカを左右に動かすループの開始と終了が逆になるようにしています．

もう1つ注意することは，イカの編隊が，

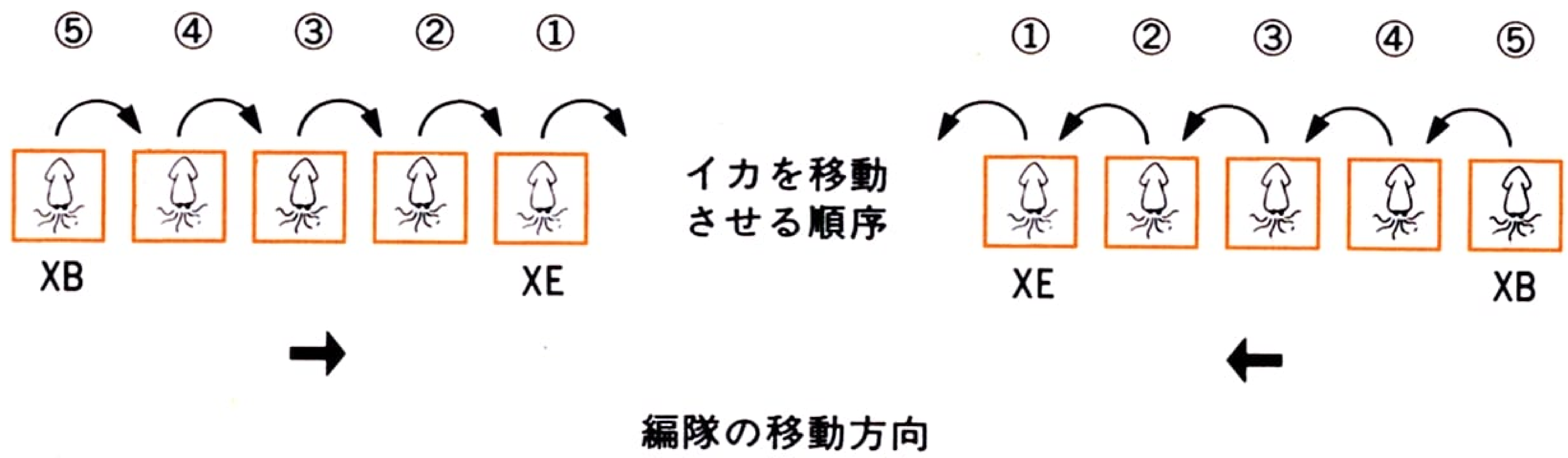
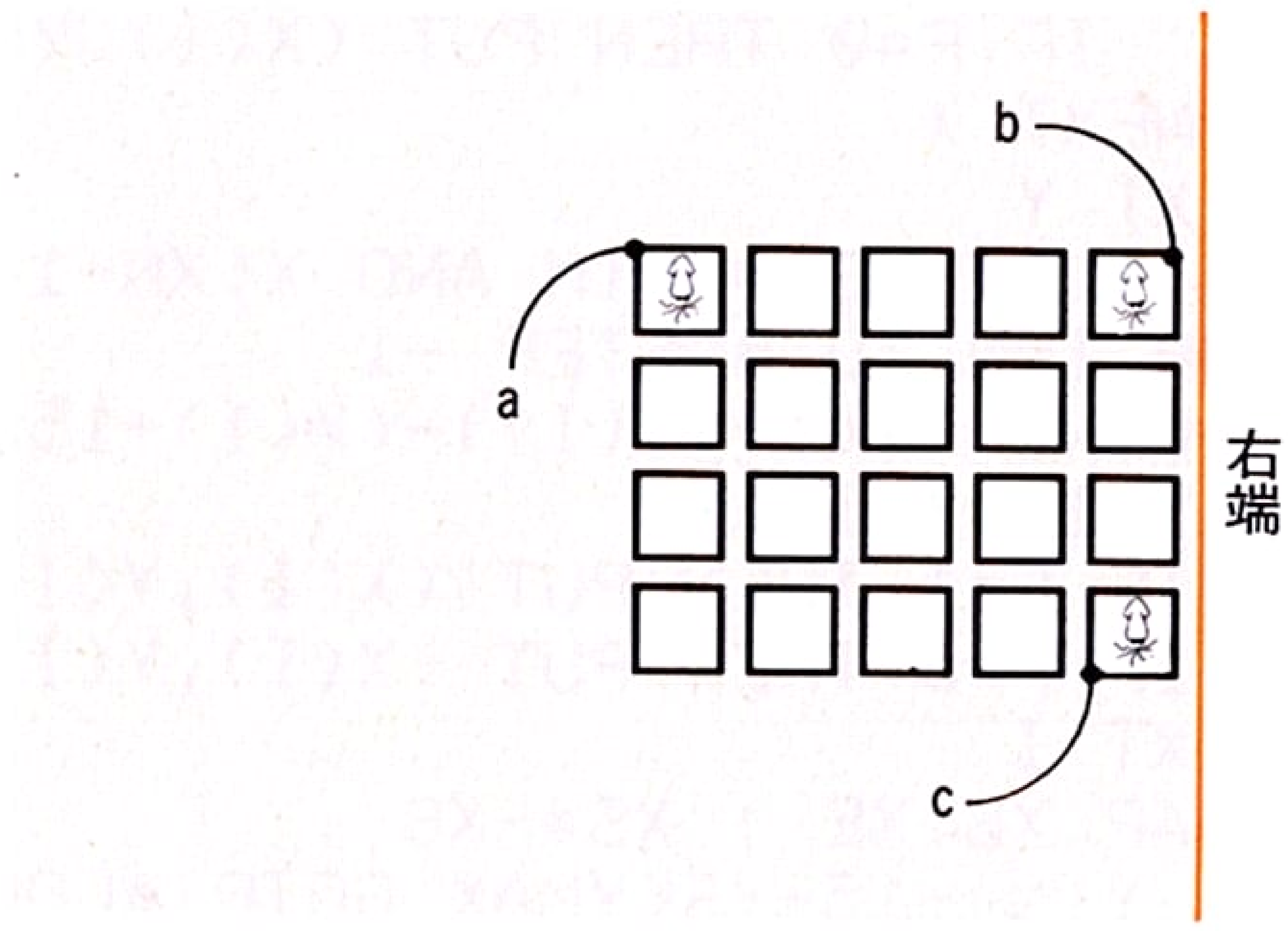


Figure 1

右端，左端あるいは下端に達したときの判定です．右端の場合は編隊の中の右端の列のイカの最も右寄りのドット，左端はその逆の最も左寄りのドット，下端は最下列のイカで判定しなければなりません(*Figure 2* 参照)．プログラム中で，右端と左端は 410 行，下端は 490 行で判定しています．

イカのキャラクタは，青い眼のイカと赤い眼のイカのデータを作り，450，460 行で PUT 命令を使って交互に出力しています．このデータを少し変えれば，昔懐かしのインベーダー・ゲームになってしまいます．



それぞれ a, b, c の点で判定する

Figure 2

List 4.12

```
100 ' List 4.12
110 '
120 XMIN=10  : YMIN=10  : XMAX=630  : YMAX=199
130 XN=4     : YN=4     : XS=45     : YS=25
140 XE=0     : F=1
150 N=XN*YN-1 : XB=XN-1
160 DIM X(N),Y(N),D1%(51),D2%(51)
170 '
180 SCREEN 0,0 : CLS : LINE (0,0)-(639,199),1,BF
190 GOSUB *RDDATA
200 FOR I=1 TO 6
210   X=INT(RND*550)-30 : Y=INT(RND*70)+130
220   FOR J=1 TO 10
```

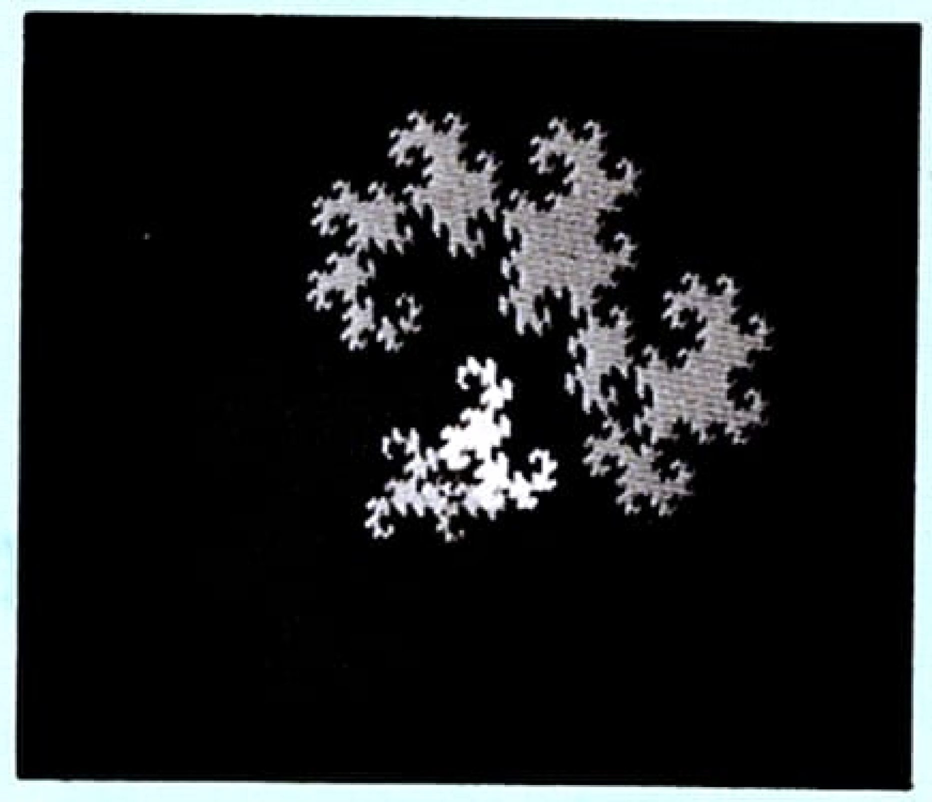
泡を描く

```
230     Y=Y-J*2 : CIRCLE (X+10*SIN(J*.6),Y),J,7
240   NEXT J
250 NEXT I
260 FOR I=0 TO N
270   X(I)=XMIN+(I MOD XN)*32 : Y(I)=YMIN+(I ¥ XN)*16
280 NEXT I
290 '
300 *LOOP
310 F=1-F : BEEP 1 : BEEP 0
320 FOR Y=0 TO YN-1
330   FOR X=XB TO XE STEP -SGN(XS)
340     I=X+Y*XN
350     LINE (X(I),Y(I))-(X(I)+15,Y(I)+15),1,BF
360     X(I)=X(I)+XS                              イカを左右に動かす
370     IF F=1 THEN PUT (X(I),Y(I)),D1%,PSET
380     IF F=0 THEN PUT (X(I),Y(I)),D2%,PSET
390   NEXT X
400 NEXT Y
410 IF X(0)+XS>=XMIN AND X(XN-1)+15+XS<=XMAX GOTO *LOOP
420 FOR I=N TO 0 STEP -1
430   LINE (X(I),Y(I))-(X(I)+15,Y(I)+15),1,BF
440   Y(I)=Y(I)+YS                               イカを1段下げるルーチン
450   IF F=1 THEN PUT (X(I),Y(I)),D1%,PSET
460   IF F=0 THEN PUT (X(I),Y(I)),D2%,PSET
470 NEXT I
480 SWAP XB,XE : XS=-XS
490 IF Y(N)+15+YS<YMAX GOTO *LOOP ELSE END
500 '
510 *RDDATA
520 RESTORE 580
530 FOR I=0 TO 51 : READ D1%(I) : NEXT I
540 RESTORE 640                                  イカのデータを読み込む
550 FOR I=0 TO 51 : READ D2%(I) : NEXT I
560 RETURN
570 '  青い眼のイカのデータ
580 DATA 16,16,-1,1,1,-1,-32765,-32765,-1,-8177,-8177,-1,-1985
590 DATA -1985,-1,-32765,-32765,-1,-16377,-16377,-1,-8177,-8177
600 DATA -1,-8177,-8177,-1,-4065,-4065,-1,-1985,-1985,-1,-1921
610 DATA -1921,-1,28700,0,-1,28700,0,-1,-23926,-23926,-1,-23478
620 DATA -23478,-1,-26574,-26574,0,0
630 '  赤い眼のイカのデータ
640 DATA 16,16,-1,0,0,-1,0,0,-1,1,1,-1,-32765,-32765,-1,-8177
650 DATA -8177,-1,-1985,-1985,-1,-32765,-32765,-1,-16377,-16377
660 DATA -1,-8177,-8177,-1,-8177,-8177,-1,-4065,-4065,-1,-1985
670 DATA -1985,-1,-897,-897,-1,640,29340,-1,-32126,-3426,-1
680 DATA -17286,-17286,0,0
```

ドラゴン曲線

再帰的なグラフィック図形のプログラムでは，サブルーチンが自分自身のみをコールするという構造をとっているものが一般的です．けれども世の中には，あるサブルーチンが別のサブルーチンをコールし，そのサブルーチンが別のサブルーチンをコールし，巡りめぐってまた自分自身が使われるという，ほとんど人間の理解のレベルを超越した構造の再帰的グラフィックが存在しています．

ここに紹介するのも，そのようなプログラムのひとつです．もっぱら左へ曲がりたがるサブルーチン(250行)と，逆に右へばかり曲がろうとするサブルーチン(320行)が，お互いに相手をコールしながらもつれあい，ある曲線を織り上げていきます．この曲線はドラゴンカーブと呼ばれ，この手のグラフィックが好きな人々の間ではとても有名なシロモノです．ドラゴン曲線には，再帰する深さによって1次，2次，…といろいろな種類がありますが，ここでは0次から18次までを連続的に描かせています．

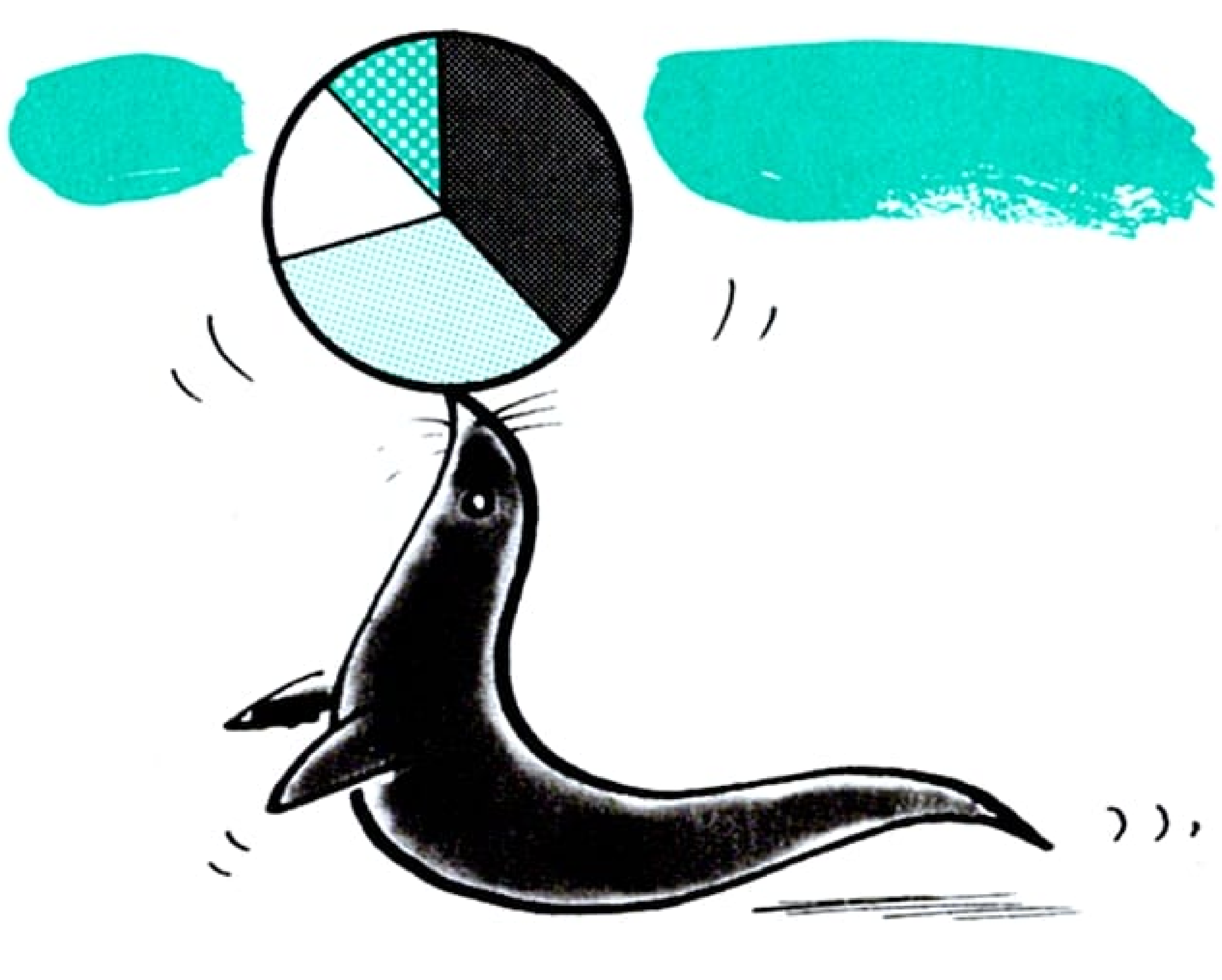


```
100 ' List Ex.4
110 '
120 DEFINT A-Z
130 SCREEN 0,0 : CLS 3
140 '
150 C=1 : MAX=18
160 XS(0)=0  : XS(1)=-2 : XS(2)=0 : XS(3)=2
170 YS(0)=-1 : YS(1)=0  : YS(2)=1 : YS(3)=0
180 '
190   POINT(320,100) : A=1 : D=0
200 *LOOP
210   GOSUB *LEFT
220   C=((C+1) MOD 7)+1 : D=D+1
230 IF D<=MAX THEN *LOOP ELSE END
240 '
250 *LEFT
260 IF D=0 THEN LINE -STEP(XS(A),YS(A)),C : RETURN
270 D=D-1 : GOSUB *LEFT
280 A=(A+3) MOD 4 : GOSUB *RIGHT
290 D=D+1
300 RETURN
310 '
320 *RIGHT
330 IF D=0 THEN LINE -STEP(XS(A),YS(A)),C : RETURN
340 D=D-1 : GOSUB *LEFT
350 A=(A+1) MOD 4 : GOSUB *RIGHT
360 D=D+1
370 RETURN
```

【5章】

グラフへのアプローチ

いろいろな関数の曲線
折れ線グラフ
円グラフ

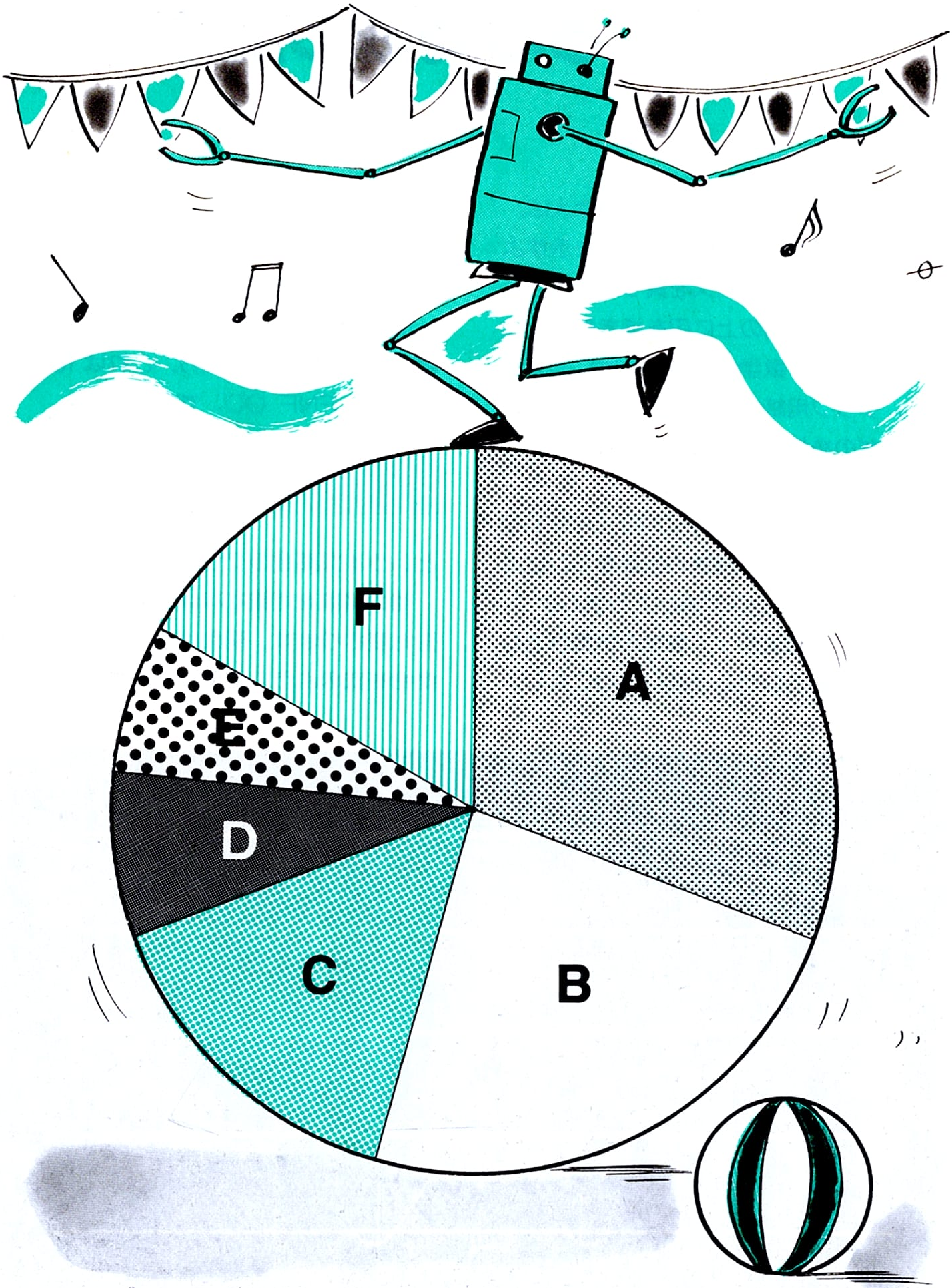


本章では，グラフの表示法について説明します．

人間の意識は，コンピュータのCPUと同様，シリアル処理を基本としているのです．だから融通がききません．けれども人間は，視覚という，パラレル処理のお手本ともいうべき有能な知覚を備えています．これがどれだけ我々の役に立っていることか．

太古の海を泳いでいたシーラカンスのような原始魚類が人間にまで進化をとげることができた背景には，脊椎動物の優秀な視覚の影響が大いにあったのだともいわれています．

この優れた視覚に訴えかけ，情報を伝える，グラフという手段は，人類の知恵であります．大いに利用したいものです．


F
A
E
D
C
B

いろいろな関数の曲線

グラフィックに興味を抱くような人は，往々にして子供の頃からその片鱗を示すようです．初めてコンパスを買ってもらい，嬉しくて家中の紙の上に円を描き散らしたり，中学では，数学の授業がやけにおもしろく，グラフ用紙の上にやたらと直線だの放物線だのを描いては喜んでいた記憶はございませんか．

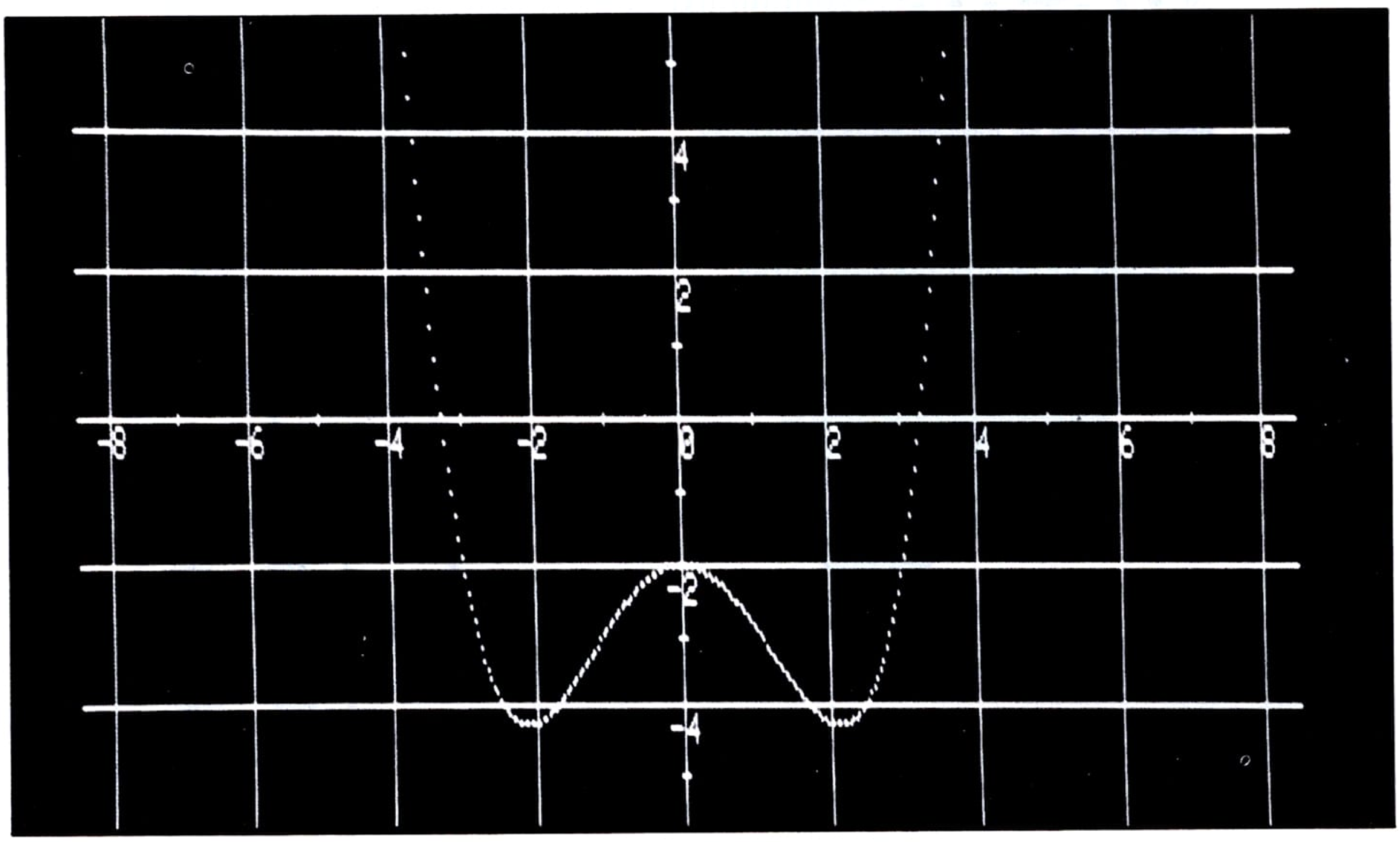
ここではグラフ用紙をPC-8801 mk IIの画面に置き換えて，関数の曲線をコンピュータ・グラフィックスで描いてみましょう．けれども，これを使って算数のお勉強をしようとか，そういうことはいいっこなし．純な心で画面に現れる曲線の美しさをめでてください．

このプログラムを実行すると，ワインボトルの底のようにも見える曲線が表示されます(所要時間は約1分)．これは180行で定義したFNF（X)，すなわち，

$$y=\frac{1}{9}x^4-x^2-2$$

という4次関数のグラフです．

他の関数を定義すれば，描かれる曲線の形も変わります．例えば180行の関数定義命令を次のように変更して，試してみてく



ださい．PC-8801 mk IIで使用できる種々の関数が，目に見える形となって現れます！

```
DEF FNF(X)=ABS(X)
DEF FNF(X)=ATN(X)
DEF FNF(X)=COS(X)
DEF FNF(X)=LOG(X)
DEF FNF(X)=SQR(X)
etc.
```

その他にも高校の教科書などが手近にある方は，開いてみるとおもしろい形の関数グラフが載っていて参考になるでしょう．

ところでこのプログラムには，関数定義以外にも設定を変更できる部分がいくつかあります．まずプログラムの150，160行で値を決めている4つの変数は，画面内のどの範囲にグラフを表示するかを決定しています．*Figure 1* に示したように，(XMIN，YMIN)を左上隅，(XMAX，YMAX)を右下隅の座標とする長方形の中がその表示範囲です．ここで150行を，

```
150 XMIN=400  :  YMIN=50
```

と変えてみると，表示範囲が右の方に移動し，なおかつ縮小されるのがわかるでしょう．

では次のような設定ならどうなるでしょうか．

```
150 XMIN=20  :  YMIN=70
```

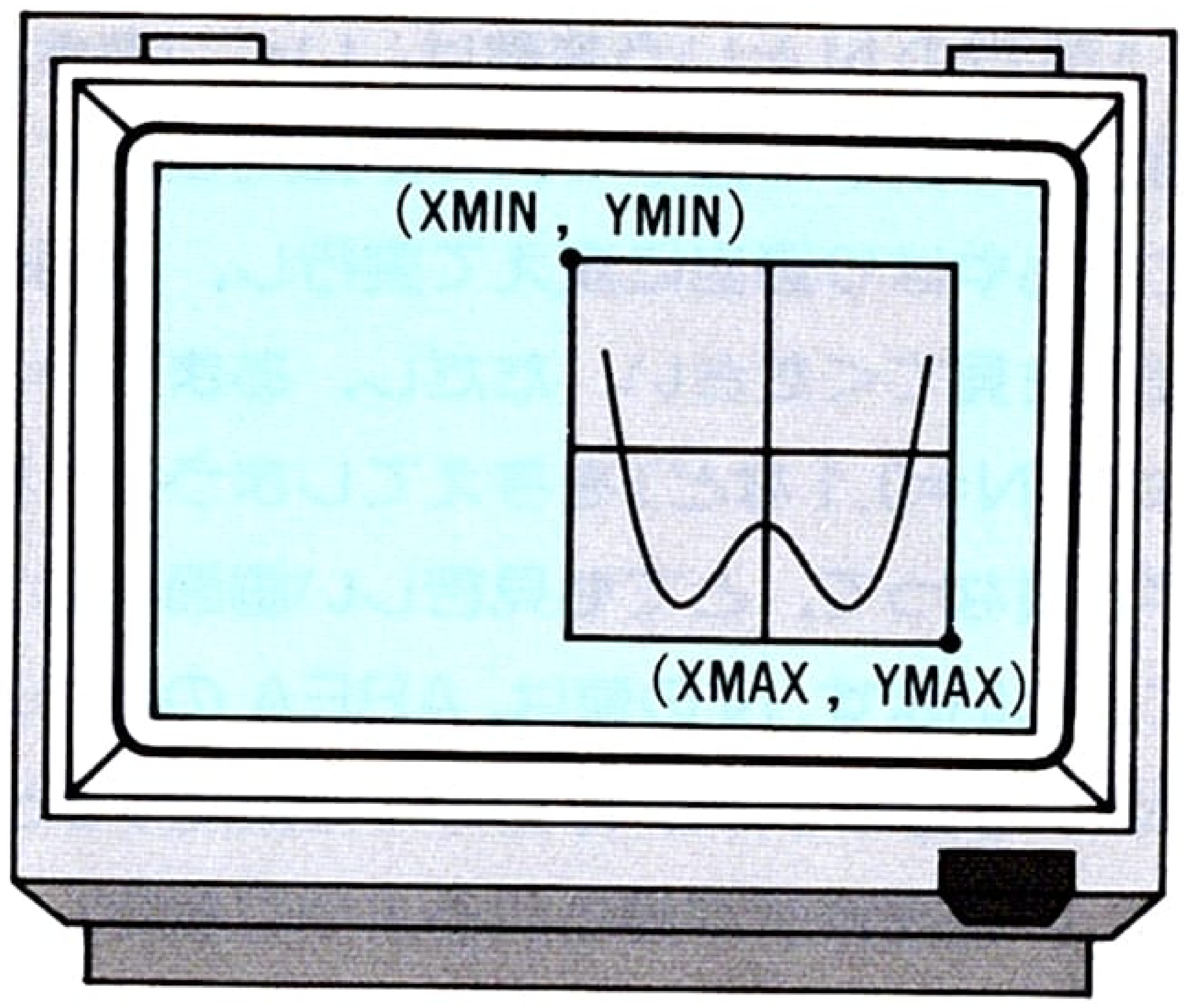


Figure 1

今度はグラフが下の方のやや平たい範囲の中に表示されるようになります．

このとき，画面上の範囲をどのように指定しても，x座標は常に約−8.5〜+8.5の範囲を表示していることに注目してください．このx座標の両端の数値を決めているのが170行のAREAという変数で，ここではこれを8.5に設定しているため，画面に表示されるのはx座標が−8.5〜+8.5という範囲になるのです．言葉ではわかりにくいとも思いますので，実際にこの値を変更した結果を見てください．例えば，170行のAREAを，

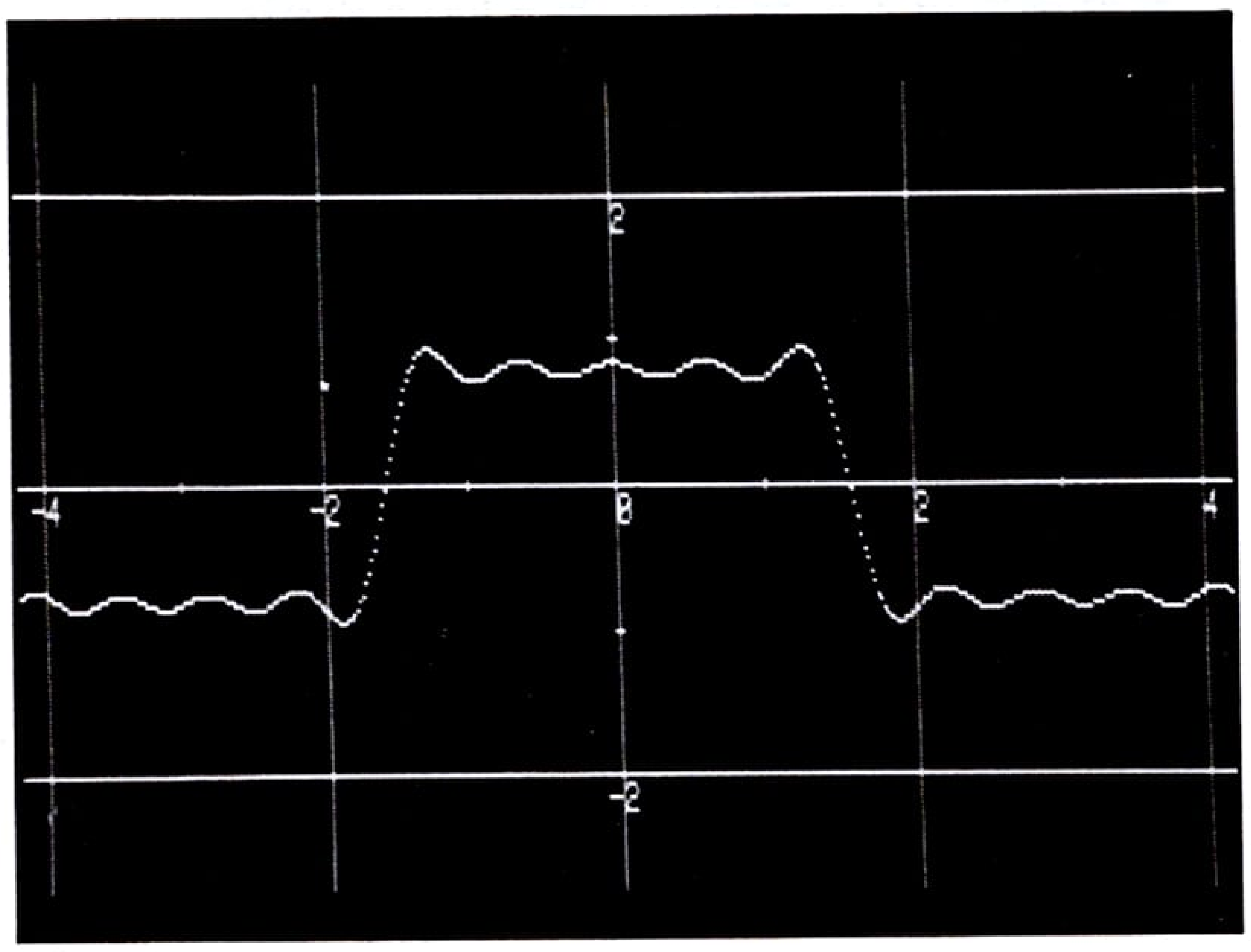
```
170 AREA=5 : N=2
```

とすれば，表示されるx座標の範囲は−5〜+5になり，その分目盛りの間隔が広がってグラフは拡大されます．なおx座標が変わると，それにつれてy座標も変わり，常に両者の1目盛りの幅が画面上で等しくなるように調整されています．

170行のNという変数は，いくつおきに目盛りの数字を振っていくか指定します．これもやはり適当に変えて実行し，画面の変化を見てください．ただし，あまり小さな値（N=0.1など）を与えてしまうと，数字が重なって，とても見苦しい画面となってしまいます．Nの値は，AREAの値のだいたい5分の1から10分の1に設定すると，美しい画面表示が得られるように設定しています．

それでは，このプログラムの全体を眺めてみましょう．このプログラムは，3つのサブルーチンから構成されています．まず1つめはプログラム中で使う各種数値の設定，2つめは座標軸や目盛りの表示，3つめは実際に点を打って，曲線を描いているサブルーチンです．

定数の設定サブルーチンは270行から始まります．この中で，X0，Y0はグラフの原点が画面上のどこに位置するのかを示す座標です．XW，YWはそれぞれグラフ表示範囲の原点から端までの距離をドット数



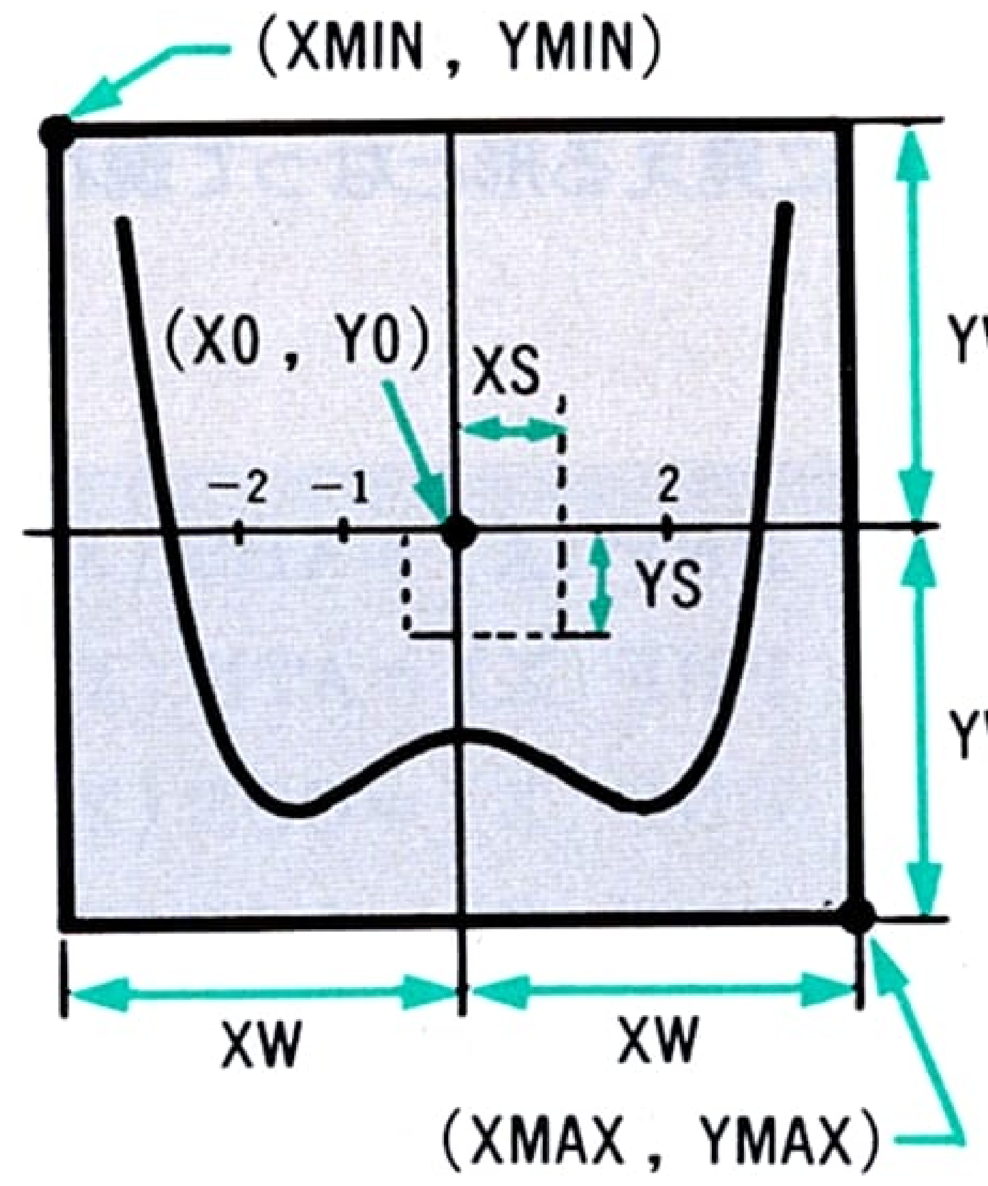


Figure 2

で表したものです（*Figure 2*参照）．関数を表示する際，1目盛りの幅をドット数で表したものをXS，YSとしています．画面上でx軸方向とy軸方向の目盛りの間隔が等しくなるよう，XSに0.5を掛けた値をYSとしています．

そこで実際の関数曲線上の点，例えば(3，2）を画面に表示するには，画面上の原点（X0，Y0）から右へXSドットの3倍，上へYSドットの2倍だけ離れた位置，つまり(X0+3 * XS，Y0−2 * YS)という座標に点を打てばよいのです．

つまり，(X，Y)という点は(X0+X * XS，Y0−Y * YS)の位置に表示するというわけです．

340行から始まるサブルーチンで，座標軸と目盛りを描きます．まずx軸とy軸を引き，そして370〜440行で各軸に目盛りをつけます．目盛りのN個ごとにケイ線を引いて数字を振るのは，450〜560行の部分です．数字はLOCATE命令とPRINT命令で表示します．LOCATE命令とPRINT命

令で表示する理由は，下地に描いてある直線を消さずに表示ができるということです．しかし，グラフィック座標とキャラクタ座標を対応させるためには，グラフィック座標を 8 で割ってやらなければなりません(ただし，WIDTH　80，25 の場合)．このプログラムでは，480，490 行，540，550 行で計算しています．

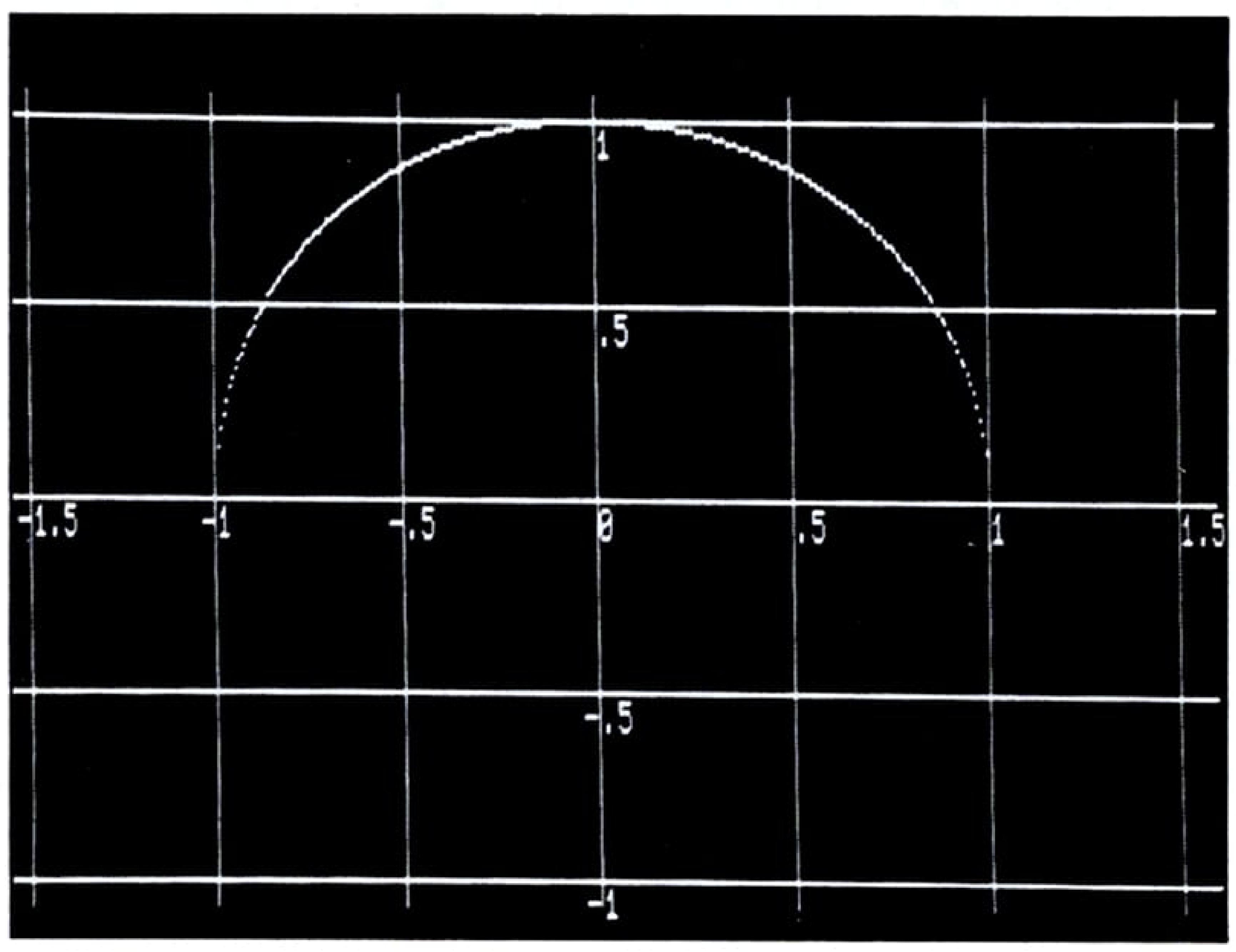
関数のグラフを描く作業を行うのが 590 行から始まるサブルーチンです．これは，X を端から端まで(－AREA～＋AREA)変化させながら，それに対する関数の値 Y を計算して点を打ちます．まず(X，Y)を画面上のグラフィック座標(GX，GY)に換算し，GY が指定した表示範囲，すなわち YMIN から YMAX の間にあることを確認した上で点を打つ必要があります．

さて，以上 3 つのサブルーチンを順に実行していけば，関数のグラフは描けます．事実，今回使った関数（$y=\frac{1}{9}x^4-x^2-2$）についていえば，ここまでで，OK です．しかし，どんな関数についてもグラフが表示できるようにするためには，さらにひとつだけ注意すべきことがあります．というのは，特定の X の値に対しては，計算ができずエラーが生じてしまう関数があるからです．例えば，ルート関数を表示しようとして，180 行で SQR(X)を定義した場合，もし何の対策も講じておかなければ，負の数に対してルートを計算したとたんにエラーが起きて，プログラムは止まってしまいます．また，値が非常に大きくなるような関数（$y=x^{100}$など）ではオーバーフロー・エラーが起きる可能性もあります．

このように，プログラムの実行が止まってしまうのを防ぐために，670 行以降のエラー処理ルーチンが必要になります．200 行で行っている設定により，プログラムのどこでエラーが起きても，670 行以降のこの部分が実行されます．

もしエラーが発生したのが 610 行または 620 行であったならば，それは関数値を計算するとき，あるいはその値をグラフィック座標に変換するときのオーバーフローか何かに違いありません．そのときは，素知らぬ顔をして 640 行に戻ってしまいます（エラーが出るほどの値は，当然グラフの表示範囲には収まらないものですから，630 行の PSET 命令は飛び越してもかまいません)．

エラー発生が 610 行と 620 行のどちらでもない場合には，本当にプログラムのどこかに間違いがあるはずです．そのような場合にはそれを表示してプログラムの実行を終了させます．これは，プログラムの打ち込み作業にはミスがつきものである，ということへの配慮です．



このように，エラー処理ルーチンを設けたときには予測したエラーと他のエラーをきちんと分けて処理しておかないと，間違っているくせに動くだけは動く，という困ったプログラムができてしまいます．くれぐれもご注意ください．

List 5.1

```
100 ' List 5.1
110 '
120 CONSOLE 0,25,0,1 : COLOR 6 : WIDTH 80,25
130 SCREEN 0,0 : CLS 2
140 '
150 XMIN=20  : YMIN=10
160 XMAX=620 : YMAX=190
170 AREA=8.5 : N=2
180 DEF FNF(X)=(1/9)*X^4-X^2-2 ……関数の定義
190 '
200 ON ERROR GOTO *ERRSUB
210 GOSUB *CONST
220 GOSUB *AXIS
230 GOSUB *GRAPH
240 ON ERROR GOTO 0
250 COLOR 7 : END
260 '
270 *CONST
280 P=.5
290 X0=(XMAX+XMIN)/2 : Y0=(YMAX+YMIN)/2
300 XW=(XMAX-XMIN)/2 : YW=(YMAX-YMIN)/2     } 定数を計算する
310 XS=XW/AREA : YS=XS*P
320 RETURN
330 '
340 *AXIS
350 LINE (XMIN,Y0)-(XMAX,Y0),1
360 LINE (X0,YMIN)-(X0,YMAX),1
370 FOR I=0 TO XW STEP XS
380   LINE (X0+I,Y0-1)-(X0+I,Y0+1),1
390   LINE (X0-I,Y0-1)-(X0-I,Y0+1),1        } x 軸，y 軸を描く
400 NEXT I
410 FOR I=0 TO YW STEP YS
420   LINE (X0-2,Y0+I)-(X0+2,Y0+I),1
430   LINE (X0-2,Y0-I)-(X0+2,Y0-I),1
440 NEXT I
```

```
450 FOR I=0 TO XW STEP XS*N
460   LINE (X0+I,YMIN)-(X0+I,YMAX),1
470   LINE (X0-I,YMIN)-(X0-I,YMAX),1
480   LOCATE (X0+I-8)/8,(Y0+2)/8 : PRINT I/XS;
490   LOCATE (X0-I-8)/8,(Y0+2)/8 : PRINT -I/XS;
500 NEXT I
510 FOR I=0 TO YW STEP YS*N
520   LINE (XMIN,Y0+I)-(XMAX,Y0+I),1
530   LINE (XMIN,Y0-I)-(XMAX,Y0-I),1
540   LOCATE (X0-8)/8,(Y0+2+I)/8 : PRINT -I/YS;
550   LOCATE (X0-8)/8,(Y0+2-I)/8 : PRINT I/YS;
560 NEXT I
570 RETURN
580 '
590 *GRAPH
600 FOR X=-AREA TO AREA STEP AREA/200
610   Y=FNF(X)
620   GX=X0+X*XS : GY=Y0-Y*YS
630   IF GY>=YMIN AND GY<=YMAX THEN PSET (GX,GY),7
640 NEXT X
650 RETURN
660 '
670 *ERRSUB
680 IF ERL=610 OR ERL=620 THEN RESUME 640
690 ON ERROR GOTO 0 : COLOR 7 : END
```

x軸に目盛りを振る (450–500)

y軸に目盛りを振る (510–560)

エラー処理ルーチン (680–690)

List 5.1a

```
170 AREA=4.2 : N=2
180 DEF FNF(X)=COS(X)-COS(3*X)/3+COS(5*X)/5-COS(7*X)/7+COS(9*X)/9
```

List 5.1b

```
170 AREA=2 : N=.5
180 DEF FNF(X)=SQR(1-X*X)
```

List 5.1c

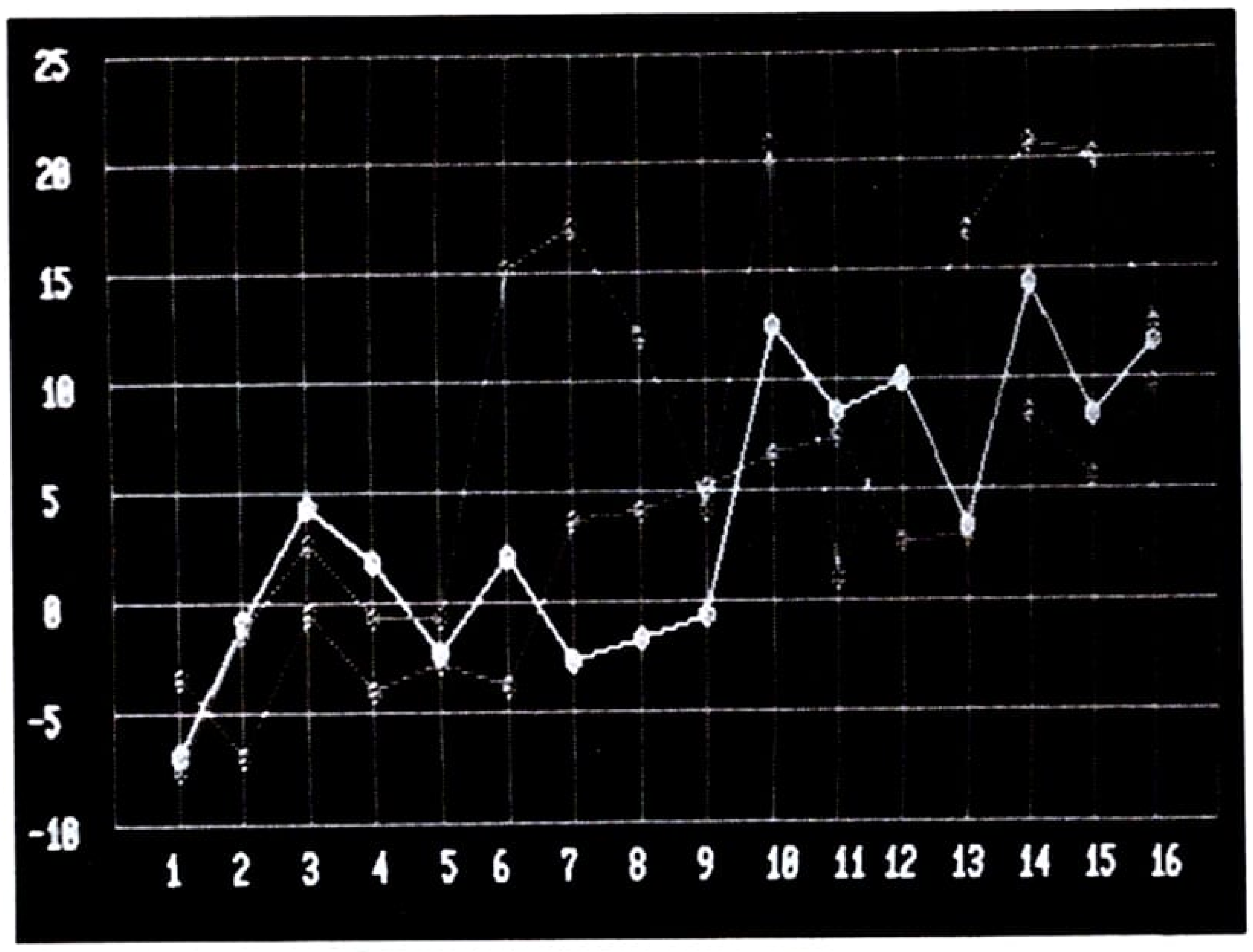
```
170 AREA=22 : N=10
180 DEF FNF(X)=SIN(X)*X/2
```

折れ線グラフ

種々のデータを蓄積し，それをうまく活用していく能力というものは，ますますその重要性を高めつつあります．プロ野球を例にとっても，動物的カンよりも，過去のデータに基づいた冷静な判断が求められる時代なのです．情報の洪水に囲まれた現代に生き残るためには，素早いデータ収集と整理能力は不可欠でしょう．

ところで，データ収集はともかく整理をするという段階になると，そのたいへんなことはいうまでもありません．だいたいデータの整理なんて作業は，数字の山との格闘になるものと相場は決まっています．

そうです，こんなときこそ，パソコンに力を発揮してもらいましょう．集計や平均などの計算をお手伝い願うのはもちろんですが，その結果の表示に際して，コンピュータのグラフィック機能はたいへん役に立ちます．そもそも，数字で表された生のデータほど理解しにくいものはありません．



しかし，これをグラフに表してみると，多数のデータをいっぺんに見渡すことができ，全体の関わり合いをはっきりと知ることができます．データの整理は，まずこの全体像をつかむことが大切．細かい数字の検討は，その後からでもよいのです．

ここに挙げたプログラムは，その１例として，データの集まりを折れ線グラフに直して表示するものです．折れ線グラフは，あるデータの時間的推移を見るには，もってこいの表示法です．これは，グラフの中では最も基本的な部類に入りますが，プログラムを作ってみると結構めんどうな処理も必要となり，思うほど簡単にはいきません．まぁ，おしなべてプログラミングなんてそういうものですけど．

ともあれ，どんなグラフが表示できるのか，プログラムを実行してみてください．これは３組のデータについて，それぞれが16回変化していく様子を見せてくれます．３組，16回，ということの数字は170行のＭおよびＮという変数で決めています．ＭやＮの値を変えると，グラフもそれに見合った形をとるようプログラミングしてありますから，これらの数値を適当に変えて，ぜひ試してみてください．

プログラムは全体として4つのサブルーチンに分けられます．上から順に，データを配列に取り込む，各種定数の設定，目盛りの表示，そして最後にグラフの表示です．では，これらを1つずつ説明していきましょう．

300行から始まるサブルーチンでは，乱数によってデータを作り出し，配列Dに格納しています．このプログラムは，あくまでも折れ線グラフのサンプルですから，このような形をとっています．しかし，実際に何らかの値を使用してグラフを描くつもりなら，この部分を他の方法，例えばINPUT命令を使うとか，外部記憶装置からデータを読み込ませるなど，それぞれの条件に合う方法に変えてデータを設定してください．

その場合，I組目のデータのJ番目の値が，D(I，J)という配列要素に入ります．例えば，3組目のデータの10番目の値は，D(3，10)となるわけです．

390〜580行は各種定数を算出するサブルーチンです．まず与えられた全データの中の最大のものと最小のものを探し出し，それぞれDMAX，DMINという変数に代入します．グラフを描く都合上，この両者の差があまり小さくては困りますので，もしDMAX−DMINが10^{-9}よりも小さい場合には，470，480行でそれぞれ10^{-9}を加減して間隔を広げておきます．

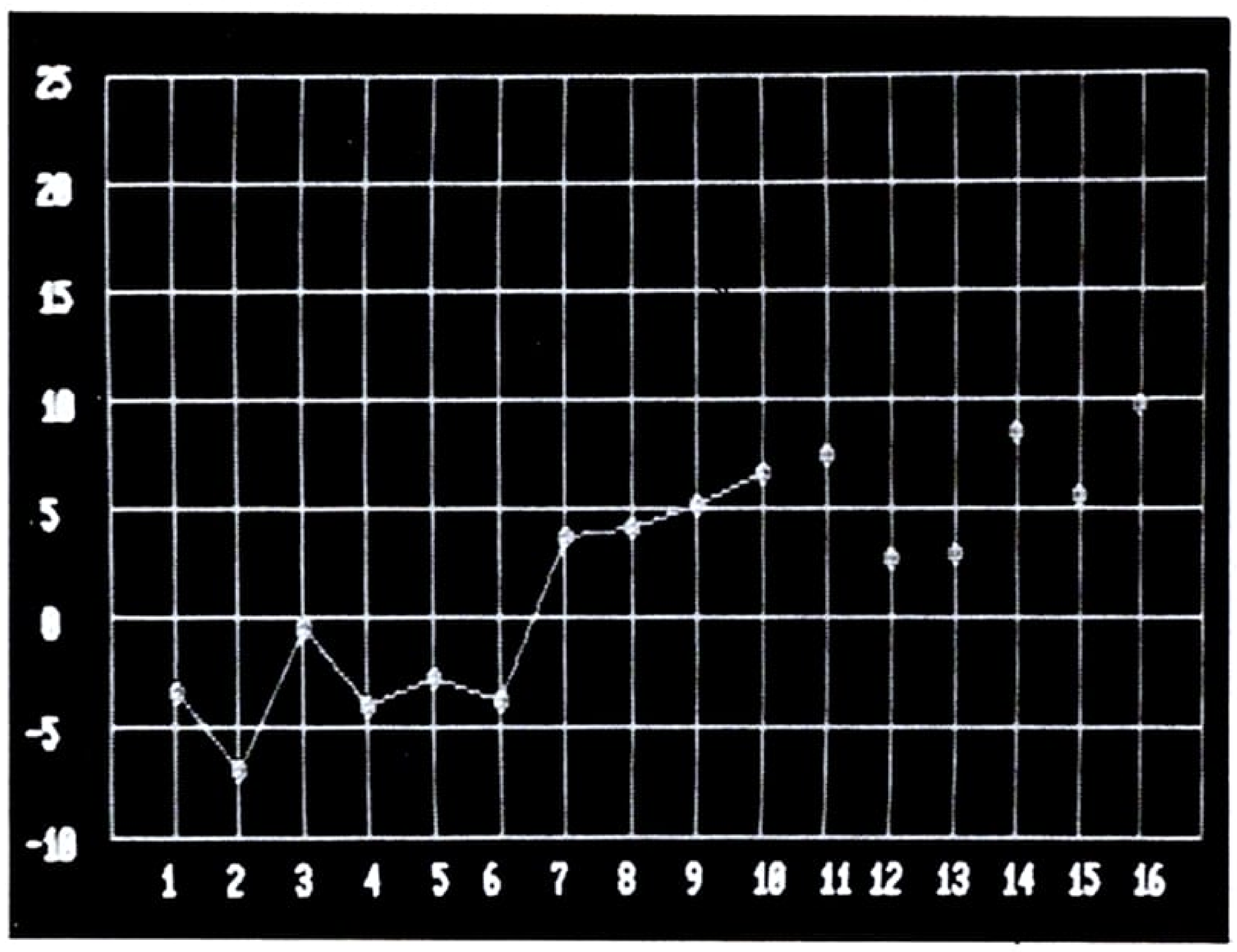
490〜510行で，データの大きさからその単位となる数値をDSに代入します．例えば，最大データDMAXと最小データDMINの差が10〜19のときはDS=2，差が20〜49のときはDS=5，差が50〜99のときはDS=10になります．

520，530行ではDSが求められた後，それを単位とした区切りのよい値に，DMAXを切り上げ，DMINを切り捨てます．

540〜580行でグラフの原点の座標(X0，Y0)を決め，x軸方向とy軸方向の単位ドット数をそれぞれ決めています．

600行からはグラフの軸や目盛りを描くサブルーチンです．コンピュータの計算に常についてまわる誤差の影響を防ぐため，2つの工夫をしています．まず620行で，Dの値をDMINからDMAXまで変化させればよいところを，わざわざDMAX+DS/2と終了地点を少し大きめにとっていること．さらに650行で，Dが十分に0に近ければそれを0としていることです．

最後のサブルーチンが，折れ線グラフの本体を描きます．まず変数Iにはあらかじめ何組目のデータを使うのか値を入れておきます．そうしてこの750行以降のサブルーチンをコールすると，I組目のデータがグラフになって表示されます．760〜790行でデ



ータの位置に印をつけ，800〜840 行でその印の間を直線で結んでいきます．グラフの色は変数 C で指定できます．

240〜270 行のループで，C の値を変えながら 750 行以降のサブルーチンを実行し，結局 M 組のデータがそれぞれ異なった色のグラフとなって表示されるわけです．

なお，このプログラムも 150，160 行の各変数の値を変えることにより，グラフを表示する領域の大きさが変更できます（*Figure 1* 参照）．

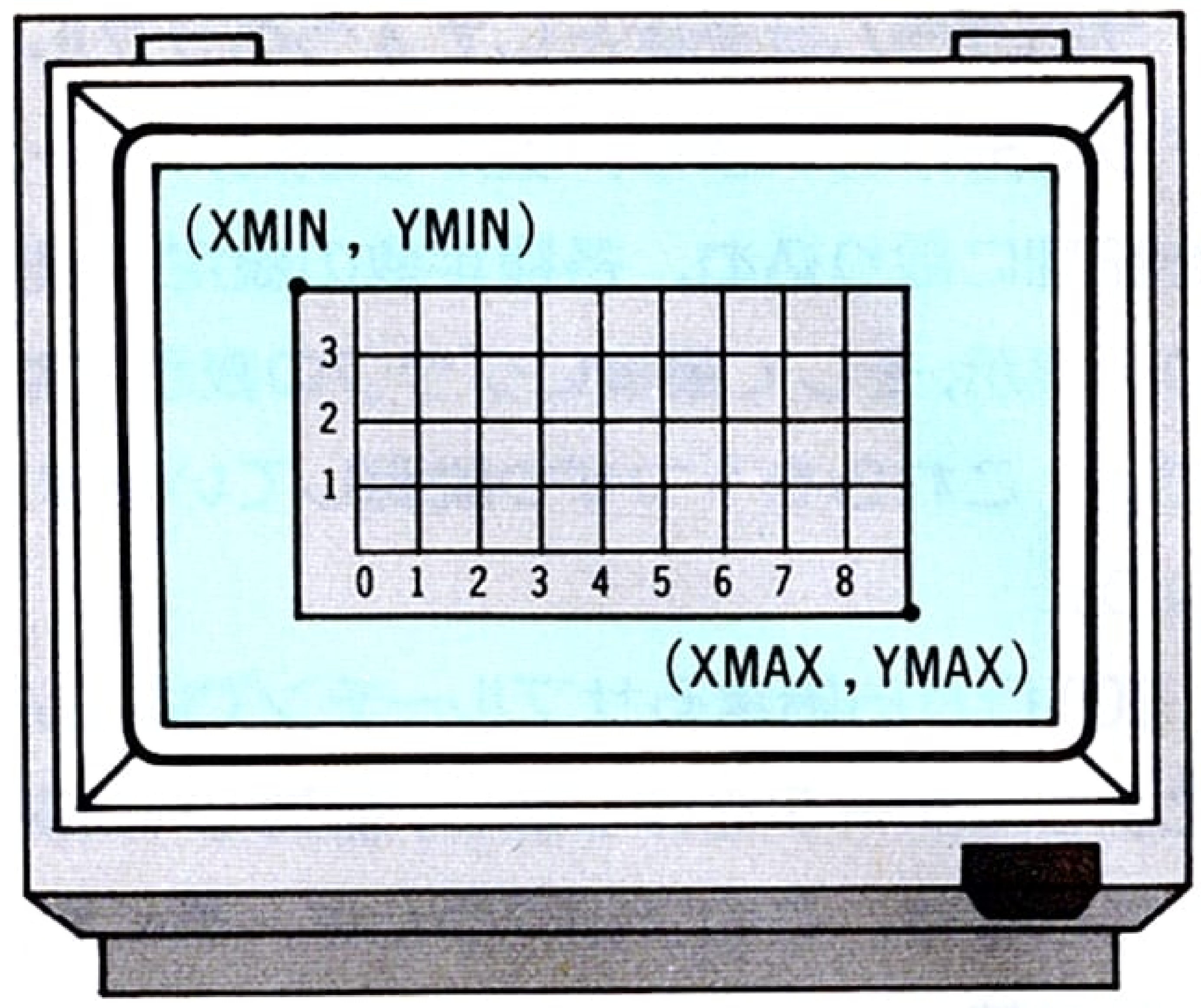


Figure 1

List 5.2

```
100 ' List 5.2
110 '
120 WIDTH 80,25 : CONSOLE 0,25,0,1 : COLOR 7
130 SCREEN 0,0  : CLS 3
140 '
150 XMIN=40  : YMIN=10
160 XMAX=600 : YMAX=190
170 M=3 : N=16
180 DIM D(M,N)
190 '
200 GOSUB *DATASET
210 GOSUB *CONST
220 GOSUB *AXIS
230 C=2 ……………………………グラフの色をパレットコード2(赤)に設定
240 FOR I=1 TO M
250   GOSUB *GRAPH
260   C=C+1 : IF C>7 THEN C=2
270 NEXT I
280 LOCATE 0,0 : END
290 '
300 *DATASET
310 RANDOMIZE VAL(RIGHT$(TIME$,2))
320 FOR I=1 TO M
330   FOR J=1 TO N
340     D(I,J)=RND*I*7+J-10 ……データを乱数で発生させる    データの設定
350   NEXT J
360 NEXT I
370 RETURN
380 '
```

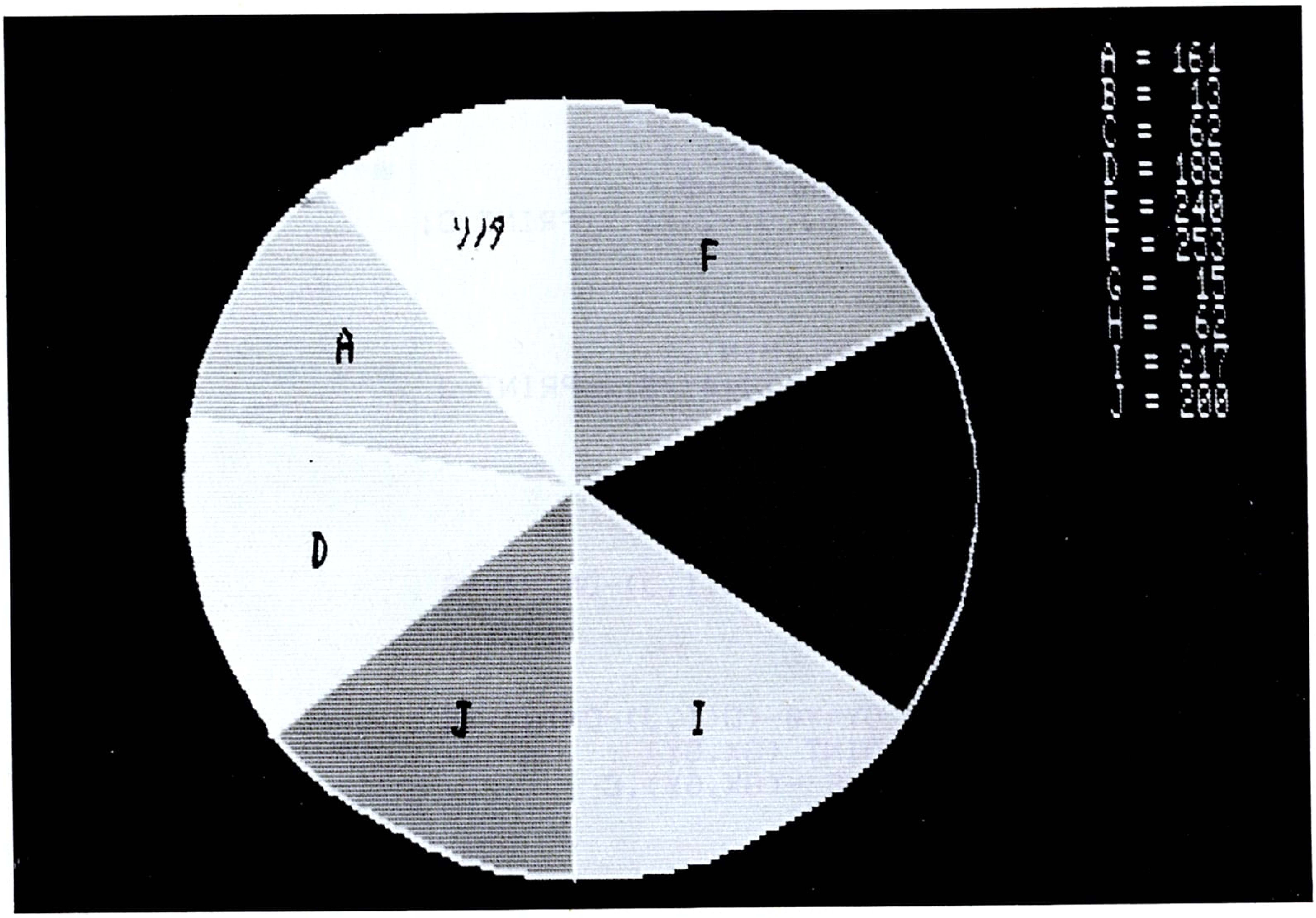
```
390 *CONST……………………………………………………定数を計算するサブルーチン
400 DMAX=D(1,1) : DMIN=D(1,1)
410 FOR I=1 TO M
420   FOR J=1 TO N
430     IF D(I,J)>DMAX THEN DMAX=D(I,J)
440     IF D(I,J)<DMIN THEN DMIN=D(I,J)
450   NEXT J
460 NEXT I
470 EPS=1E-09
480 IF DMAX-DMIN<EPS THEN DMAX=DMAX+EPS : DMIN=DMIN-EPS
490 DS=10^INT(LOG(DMAX-DMIN)/LOG(10))
500 IF (DMAX-DMIN)/DS<2 THEN DS=DS/5
510 IF (DMAX-DMIN)/DS<5 THEN DS=DS/2
520 DMAX=INT(DMAX/DS)*DS+DS
530 DMIN=INT(DMIN/DS)*DS
540 X0=XMIN+40
550 Y0=YMAX-12
560 XS=(XMAX-X0)/(N+1)
570 YS=(Y0-YMIN)/(DMAX-DMIN)
580 RETURN
590 '
600 *AXIS……x軸，y軸を描くサブルーチン
610 LINE (X0,Y0)-(XMAX,YMIN),1,B
620 FOR D=DMIN TO DMAX+DS/2 STEP DS              ┐
630   GY=Y0-(D-DMIN)*YS                          │
640   LINE (X0,GY)-(XMAX,GY),1                   │ 横ケイを描く
650   IF ABS(D)<DS/2 THEN D=0                    │
660   LOCATE (X0-40)/8,(GY-2)/8 : PRINT D;       │
670 NEXT D                                       ┘
680 FOR J=1 TO N                                 ┐
690   GX=X0+J*XS                                 │
700   LINE (GX,Y0)-(GX,YMIN),1                   │ 縦ケイを描く
710   LOCATE (GX-12)/8,(Y0+4)/8 : PRINT J        │
720 NEXT J                                       │
730 RETURN                                       ┘
740 '
750 *GRAPH
760 FOR J=1 TO N
770   GX=X0+J*XS : GY=Y0-(D(I,J)-DMIN)*YS
780   CIRCLE (GX,GY),3,C
790 NEXT J
800 FOR J=1 TO N
810   GX=X0+J*XS : GY=Y0-(D(I,J)-DMIN)*YS
820   IF J=1 THEN POINT (GX,GY)
830   IF J>1 THEN LINE -(GX,GY),C
840 NEXT J
850 RETURN
```

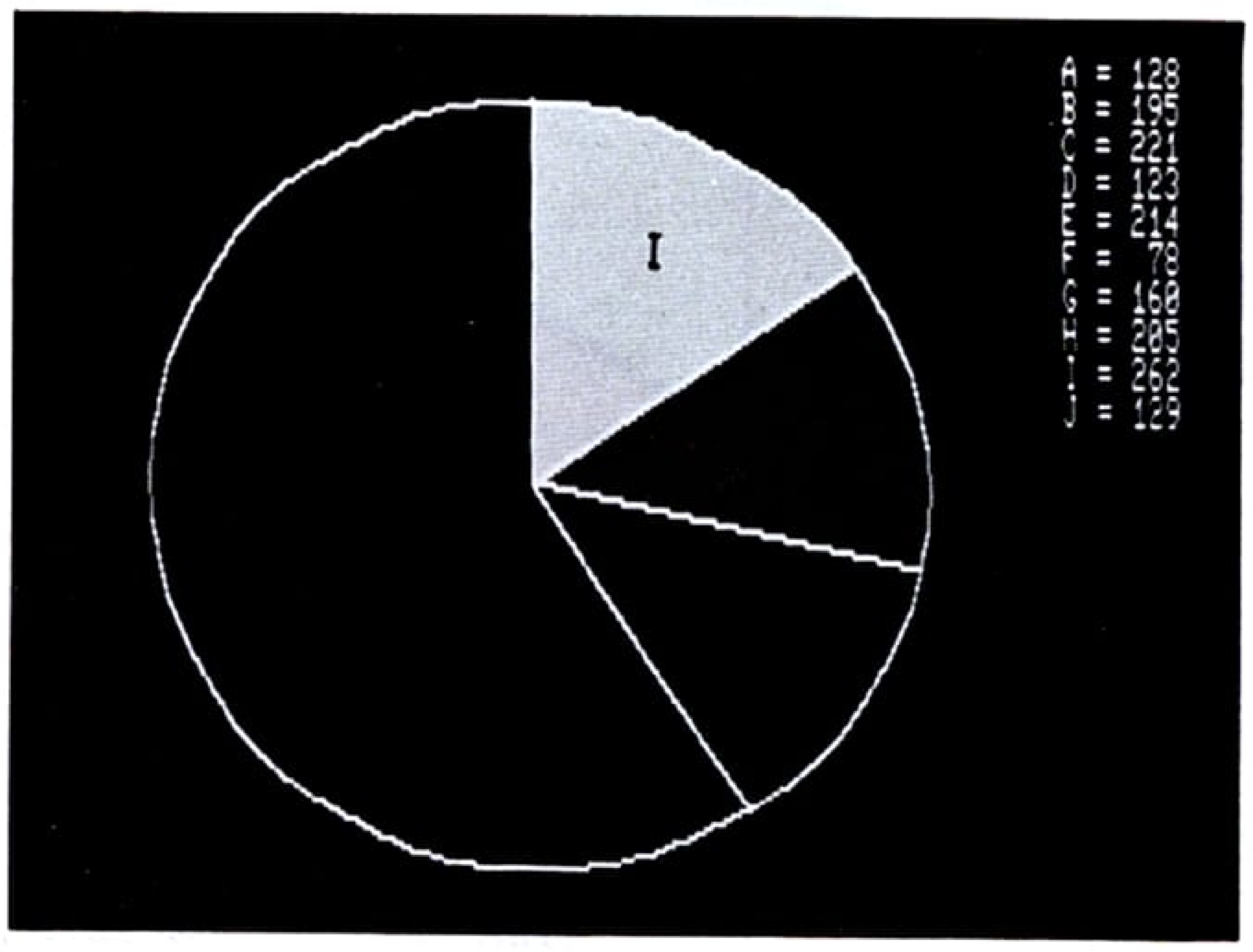
円グラフ

折れ線グラフが，主として単一のデータが時間とともに変化する様子を見るためのものであるのに対し，ある時点における多数のデータの量的関係（どれが多いか少ないか）を知る目的で使われるグラフが，この円グラフです．

聞くところによれば，人間が角度を識別する能力は，なかなか優れたものだそうです．例えば100cmと99cmの棒をしばらく見せておいたあとに，その一方を出して，さあこれは何cmの方だ，と質問される場合と，同じことを100度と99度の角度に切ったボール紙について行う場合では，人にもよるでしょうが，たいていは後者の正解率の方が高いそうな．

時計は何といっても針が回るアナログ式でなきゃね，という人が多いのも，やはり時間を角度に換算するという方式が，どこ





か心にしっくりくるためではないでしょうか．円グラフによる表示がデータの大小関係をつかみやすいというのも，同じような理由なのでしょう．

さて，このプログラムは3つに分かれています．始めの部分は配列にデータを取り込む部分ですが，ここでは乱数を使って簡単にすませています．何か実際のデータを入れてみようという場合には，配列N$に項目の名前，配列Dにそれぞれのデータを入力してください．そしてデータの個数は変数Nに格納しておきます．

300行からのサブルーチンは，グラフを表示する前のデータ処理を行います．円グラフは，全体に対する個々のデータの比率を表示するものですから，まずデータの総和を求めておかなければなりません．これを計算するのが310〜350行の部分で，ここで同時にデータの項目とその数量を画面に表示する作業も行います．

次に円グラフに特有の処理として，データを大きい順に並べ換える（ソートする）必要があります．これは370〜410行で行っています．

並べ換えが終わったなら，あまり小さなデータは表示するとかえって見づらくなってしまうので，“その他”という項目にまとめてしまいます．データの大きさがいくつ以下の場合に“その他”に入れるかというのは，いろいろ論議されてしかるべき問題とも思いますが，このプログラムでは，全体に対する比率D(I)/SUMが1/(N+5)より小さいことを規準にしています．

そして最後に，510行以降のサブルーチンで円グラフを描きます．560行のTBは，1つの扇形の描き始めの角度を示す値で，始めは0に設定しておきます．580行ではI番目のデータD(I)が，全体SUMに対してどれだけの比率なのかを計算し，そこに2πを掛けて角度（単位はラジアン）に変換します．この値をTBに足すと，I番目のデータに相当する扇形の終わりの位置TNが得られます（*Figure 1* 参照）．590〜610行で，角度TNの位置と中心を結びます．次に扇

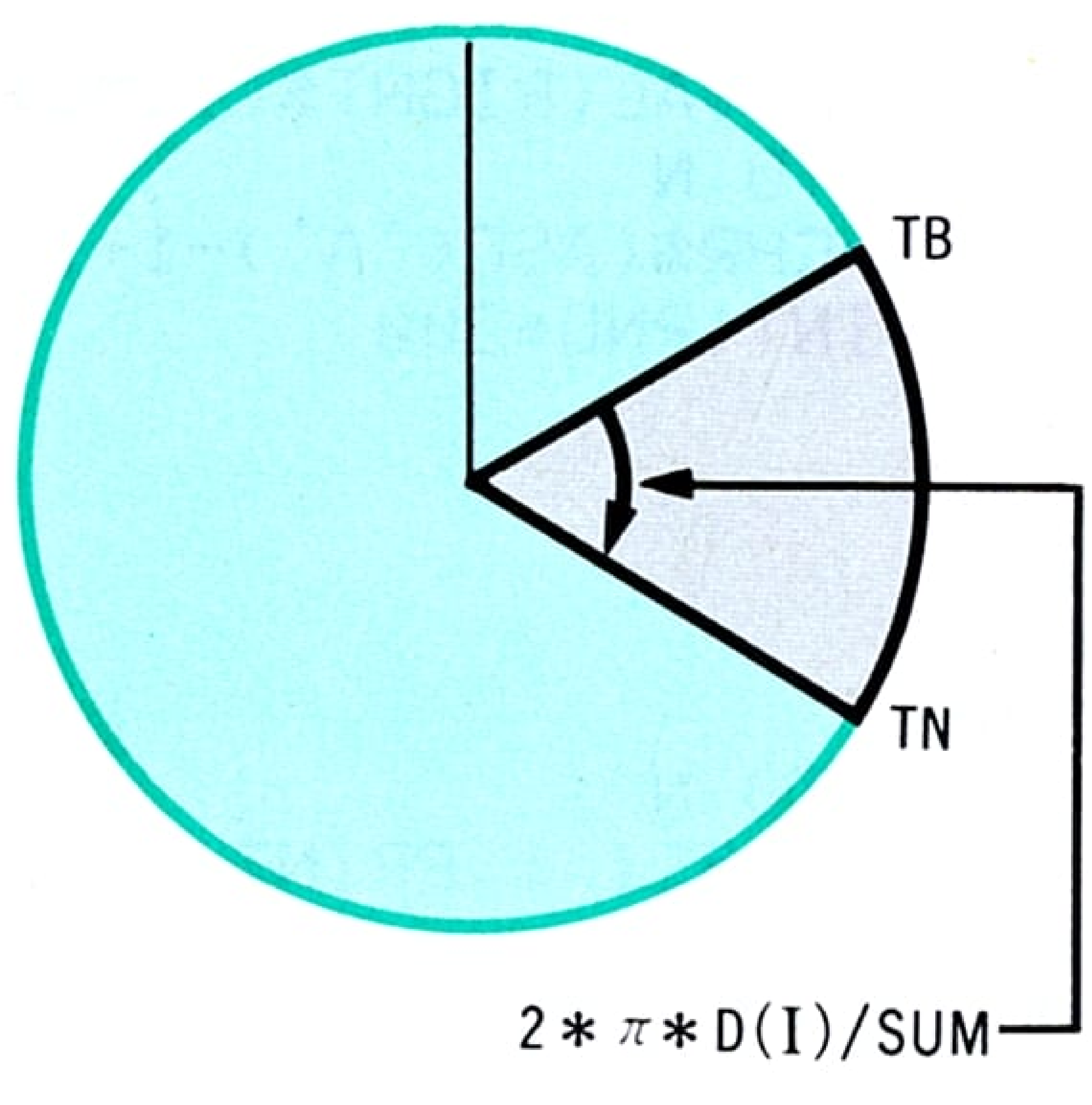


Figure 1

形の中をPAINT命令で塗りつぶします．塗りつぶしの始点として指定するのは，TBとTN間の角度で，中心から半径2/3離れた所です（*Figure 2* 参照）．この計算は，620，630行で行います．また，その同じ位置に650行のLOCATE命令とPRINT命令で項目名を書き込み，扇形を1つ描き終えます．

こうして次々と扇形を塗っていき，円グラフは完成します．

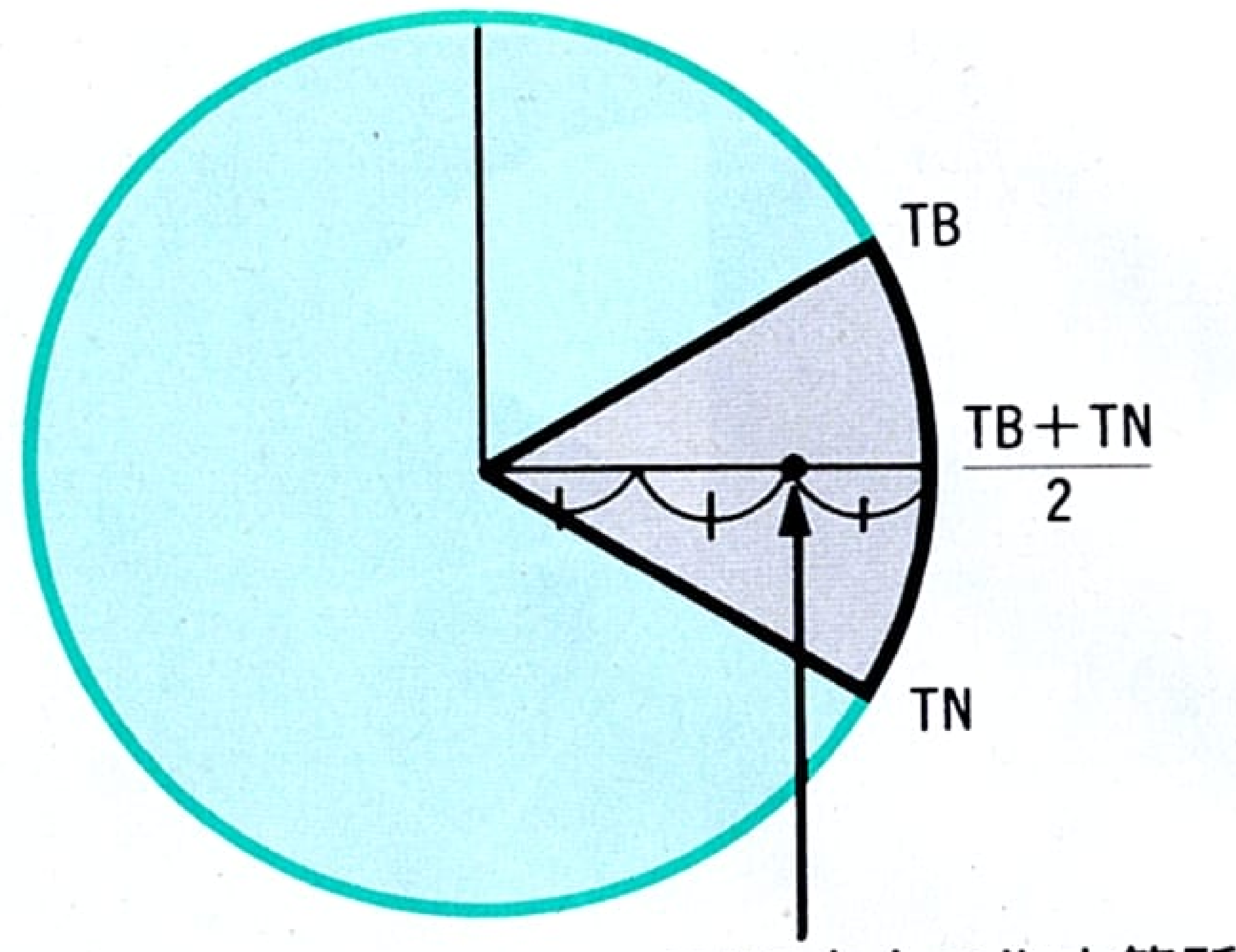


Figure 2

List 5.3

```
100 ' List 5.3
110 '
120 WIDTH 80,25 : CONSOLE ,,0,1 : COLOR 7
130 SCREEN 0,0  : CLS 3
140 '
150 N=10 : DIM N$(N),D(N)
160 '
170 GOSUB *DATASET
180 GOSUB *SORT
190 COLOR 0 : GOSUB *GRAPH
200 COLOR 7 : LOCATE 0,0 : END
210 '
220 *DATASET
230 RANDOMIZE VAL(RIGHT$(TIME$,2))
240 FOR I=1 TO N
250   N$(I)=CHR$(ASC("A")-1+I)
260   D(I)=INT(RND*300)
270 NEXT I
280 RETURN
290 '
300 *SORT
310 SUM=0
320 FOR I=1 TO N
330   LOCATE 65,I : PRINT USING "@ = ###";N$(I),D(I);
340   SUM=SUM+D(I)
350 NEXT I
```

データを乱数で発生させる（250〜260行）

データの設定（220〜280行）

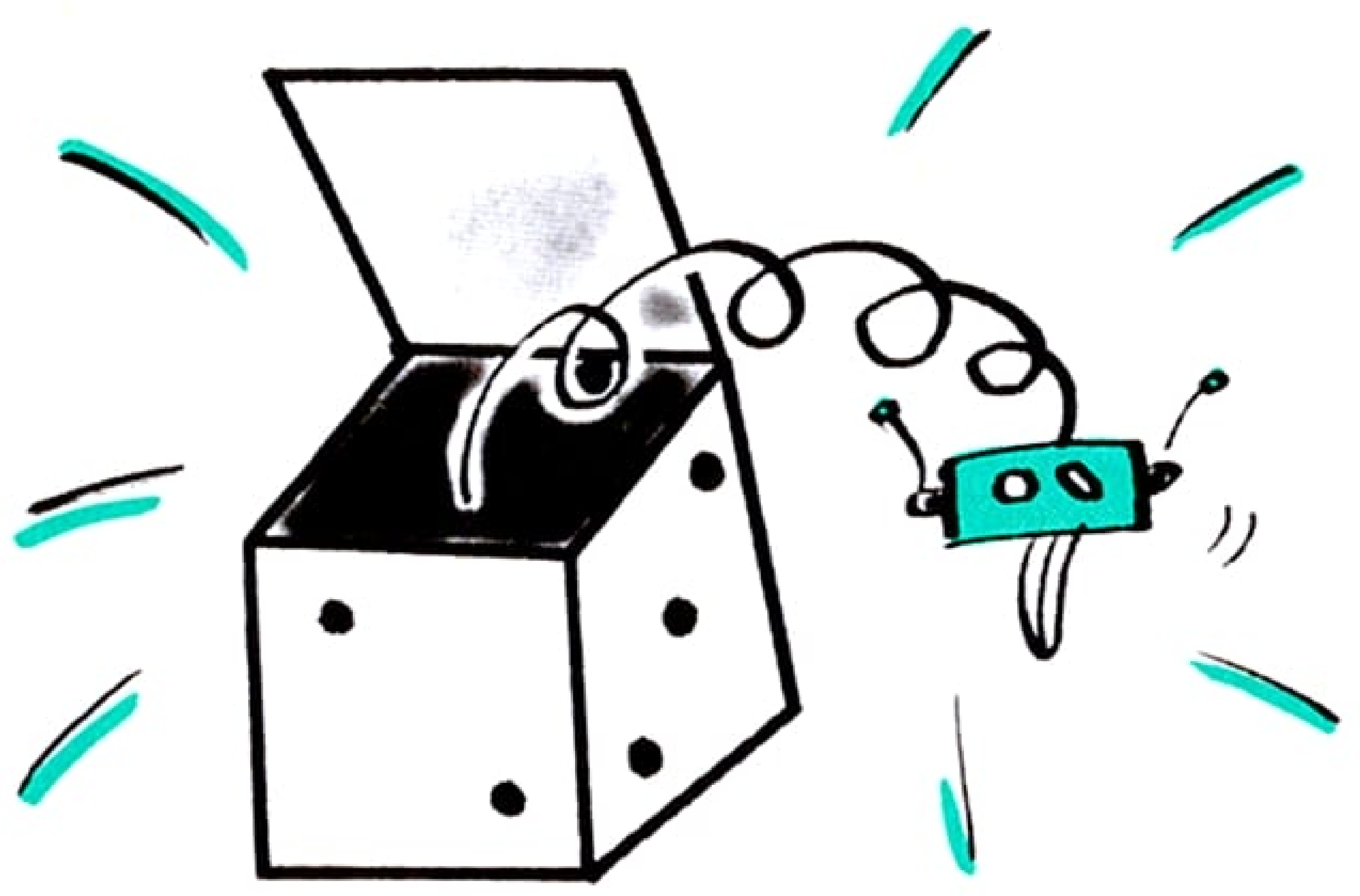
データの総和を計算する（330〜350行）

```
360 '
370 FOR J=1 TO N-1
380   FOR I=J+1 TO N
390     IF D(I)>D(J) THEN SWAP D(I),D(J) : SWAP N$(I),N$(J)    データのソート
400   NEXT I
410 NEXT J
420 '
430 TA=0 : I=N : NMAX=N
440 IF D(I)/SUM>=1/(N+5) GOTO 480
450   TA=TA+D(I)
460   I=I-1                                                   少数のデータをまとめて"ソノタ"とする
470 GOTO 440
480 IF TA>0 THEN NMAX=I+1 : N$(NMAX)="ソノタ" : D(NMAX)=TA
490 RETURN
500 '
510 *GRAPH……円グラフを描くルーチン
520 R=180 : P=.5 : PI=3.14159
530 CIRCLE (280,100),R,7
540 LINE (280,100)-(280,100-R*P),7
550 '
560 TB=0 : C=1
570 FOR I=1 TO NMAX
580   TN=TB+2*PI*D(I)/SUM
590   X=280+R*SIN(TN)                                         扇形を描く
600   Y=100-R*COS(TN)*P
610   IF I<NMAX THEN LINE (280,100)-(X,Y),7
620   X=280+R*(2/3)SIN((TN+TB)/2)
630   Y=100-R*(2/3)*COS((TN+TB)/2)*P                          扇形を塗りつぶす始点
640   PAINT (X,Y),C,7
650   LOCATE (X-LEN(N$(I))*4)/8,(Y-4)/8 : PRINT N$(I);……項目名の書き込み
660   TB=TN
670   C=C+1 : IF C>7 THEN C=1
680 NEXT I
690 RETURN
```

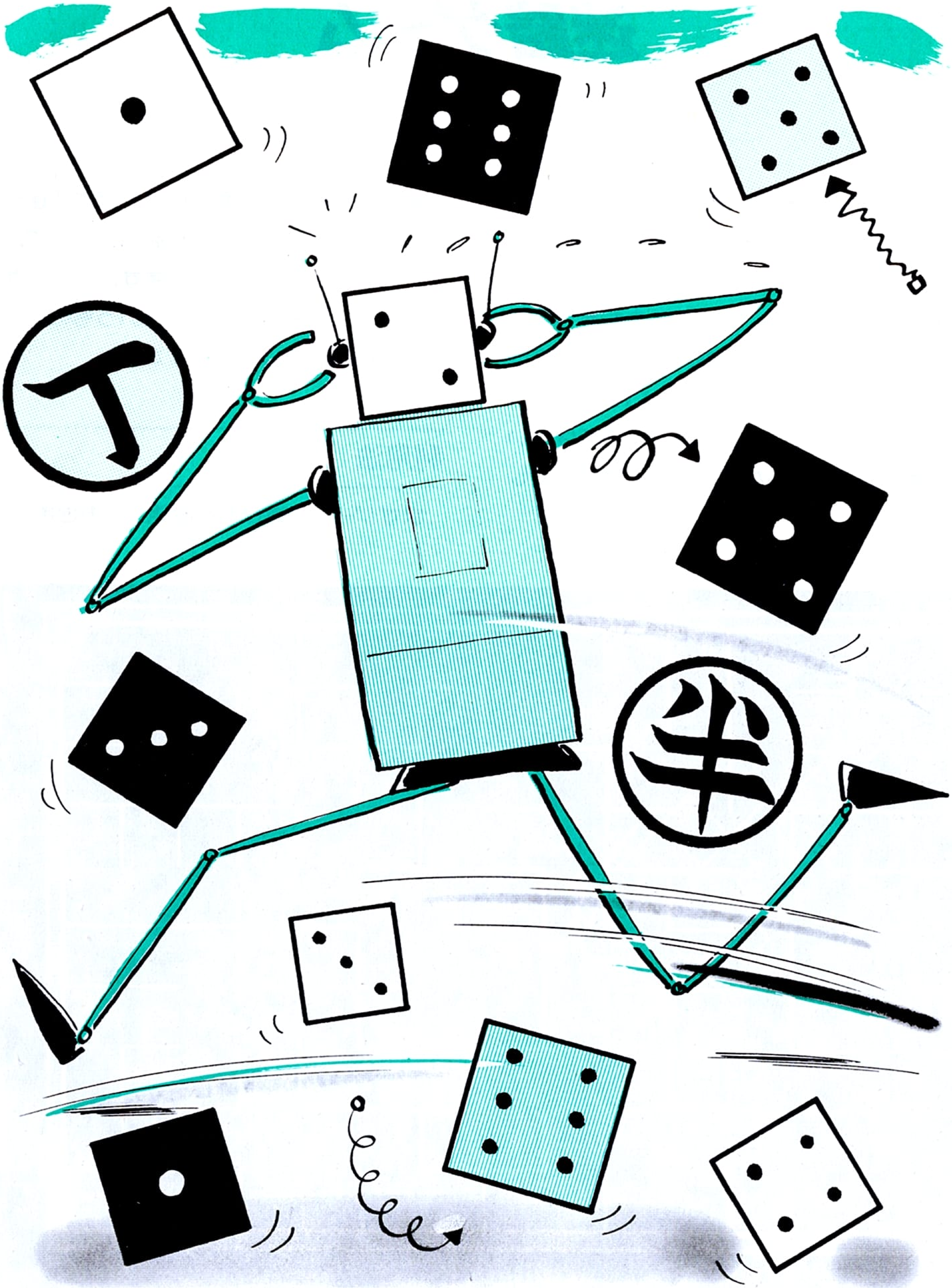
【6章】

ランダムアプリケーション

迷路ゲーム
バイオリズム
STAR WARSゲーム
焼けてないトタン屋根の上のネコ



本書をここまで読んできた方は，TVゲームやコンピュータ・グラフィックスを見たときに，およそどんなプログラムになっているか見当がつくはずです．しかし，いざ自分でプログラムを作ろうと思うと，今までのひとつひとつのブロックを，どうやって組み合わせていけばよいのか，どうもよくわからないという人もいることと思います．そこで，あくまでもサンプルとしてですが，簡単なゲームを実際に作ってみることにしました．しかし，プログラムは自分で作ってこそ上達するものです．これら既存のゲームで遊ぶだけではなく，自分でも，プログラムを作ってみてください．また，このプログラムを自分なりに変更してみるのも，プログラミング上達には有効な方法でしょう．


丁
半

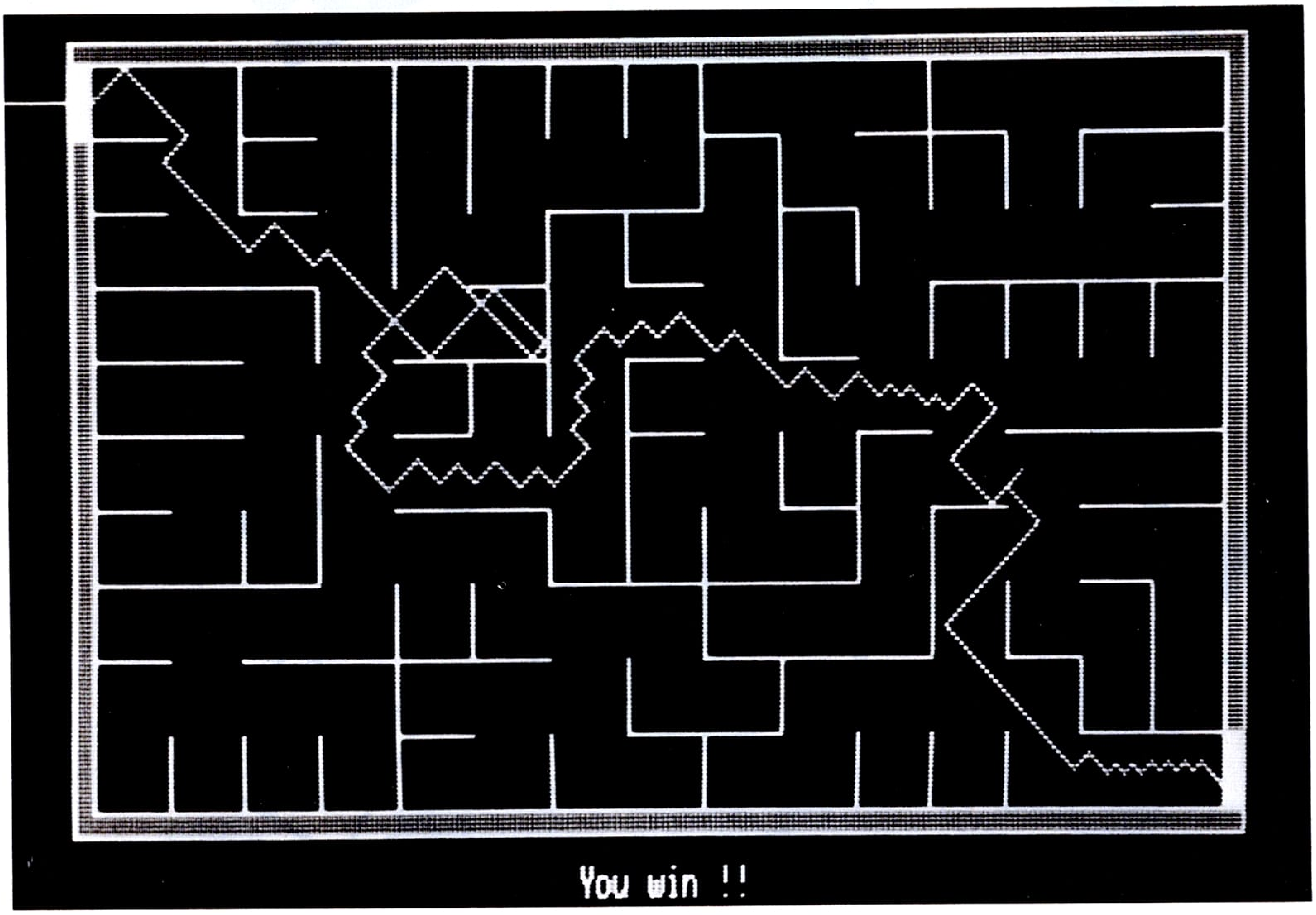
迷路ゲーム

4章に挙げられているプログラムは，ゲームのある1部分，1要素のみを取り出したものでした.「あんな画面はゲームセンターで見たことがあるぞ」というものもあったと思います．しかし，実際，自分でプログラムを組んでみようと思うと，いったいどうすればよいのか，まず何から始めればよいのか，よくわからないという人もいるでしょう．

そこで，実際に4章で取り上げた「迷路を作る」というプログラムをもとにして，ゲームを作ってみました．まず，これを参考にして，ゲームの作り方というものを，じっくりと研究してみてください．

利用法

まずゲームの遊び方を説明しておきます．



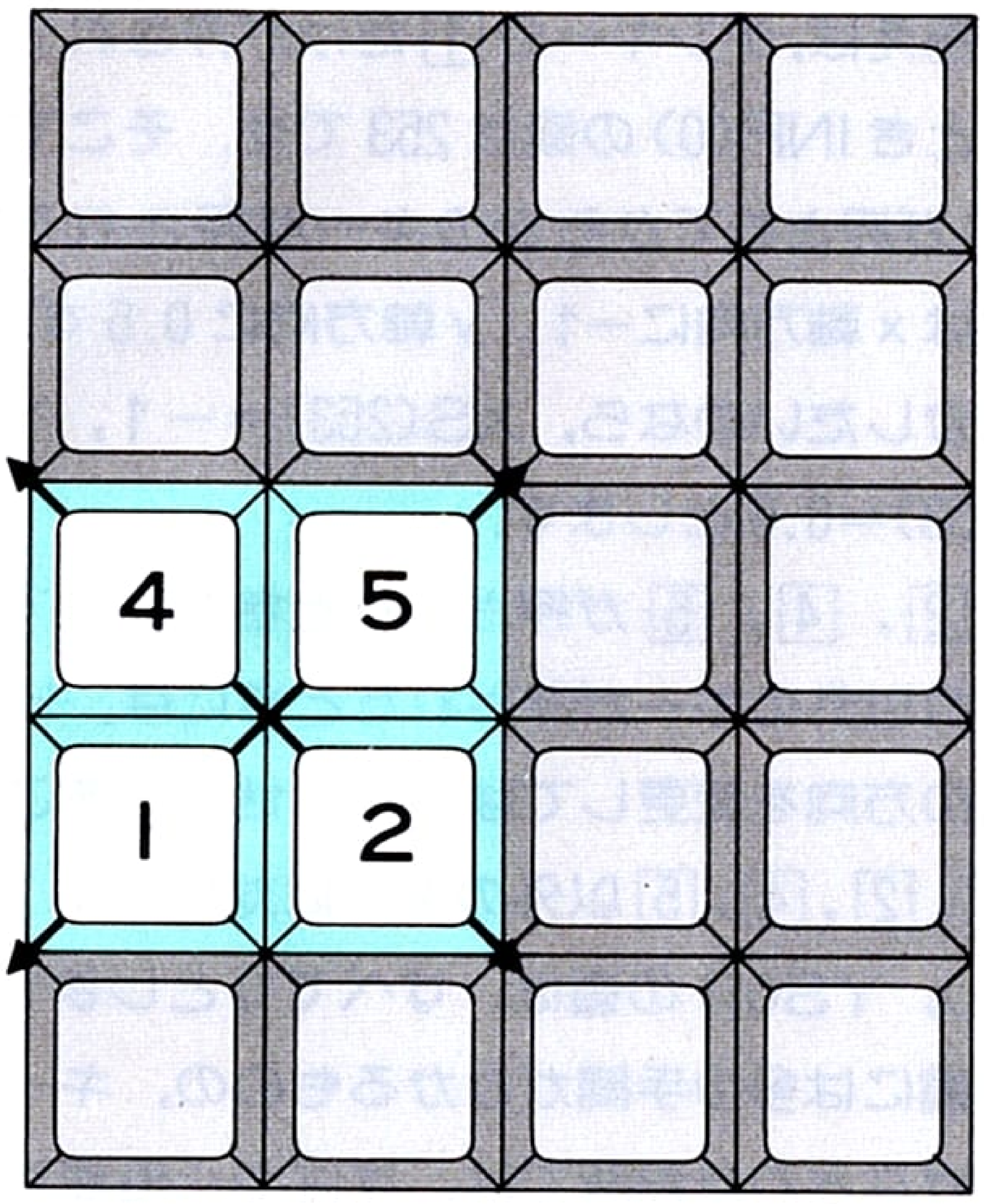


Figure 1

このゲームは，迷路の入口から，常に斜めに移動するビームを操作して，出口へ抜けていくものです．ENERGY は，100 が最大で，ビームが壁にぶつかるたびに 1 ずつ減り 0 になったら GAME OVER です．キー操作はテンキーを使い，[5] で右上に，[2] で右下に，[4] で左上に，[1] で左下にビームが進みます（*Figure 1* 参照）．

プログラムの説明

このプログラムは，4 章の「迷路を作る」というプログラムをそのまま利用して，発展させたものです．

それではリストを見ながらプログラムの説明をしましょう．1060〜1210 行は，ビームを移動させるメイン・ルーチンなのですが，ひとまず他のサブ・ルーチンをさきに説明します．

● 1230〜1540 行

これが，「迷路を作る」プログラムの本体です．このプログラムは 4 章で示したものに，1240〜1260 行の初期設定と，1540 行の RETURN をつけ加えたことの他は変更していません．これは，いうまでもなく 1030 行から呼び出されるサブルーチンとなっています．

● 1560〜1790 行

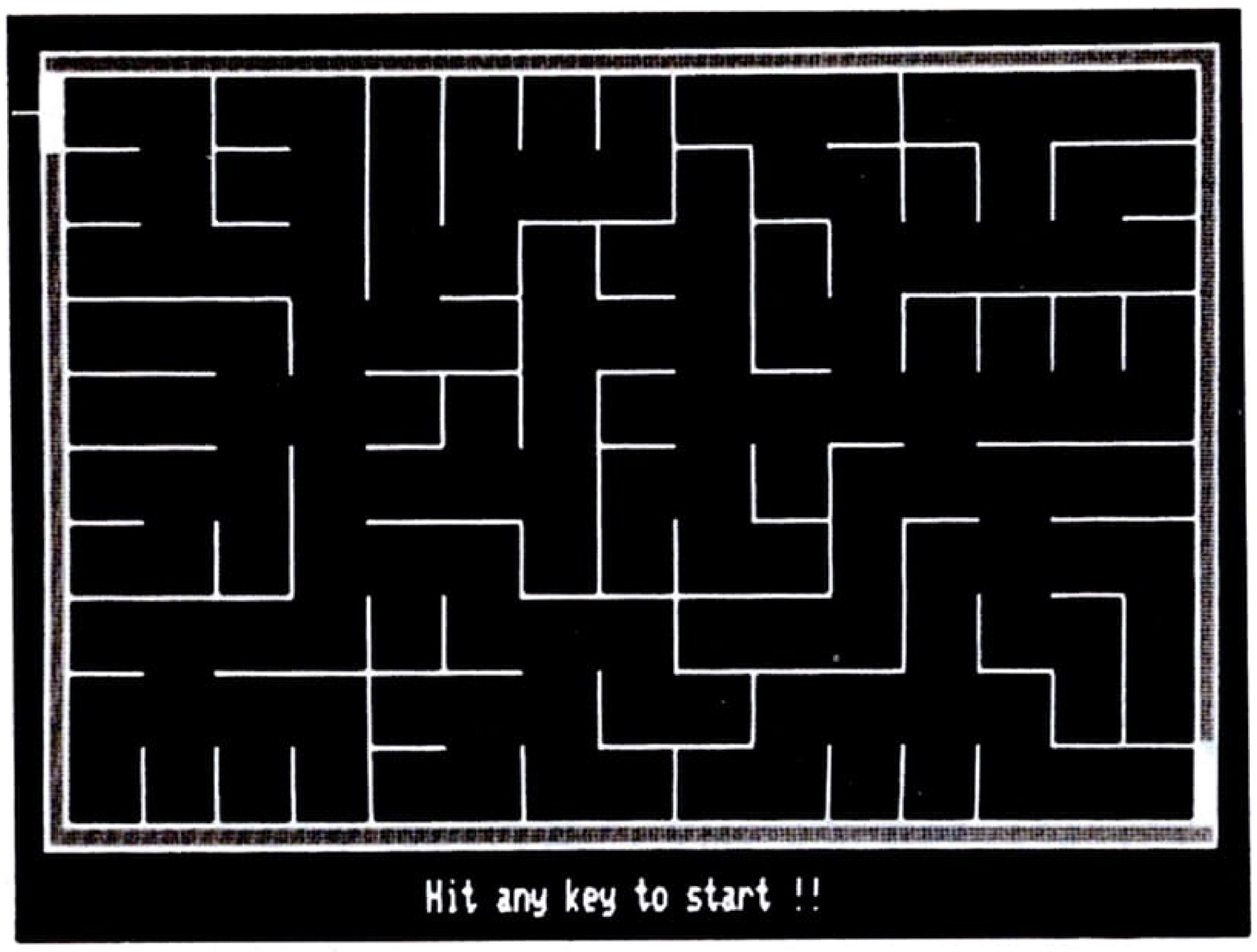
ここに，今までに出てこなかったテクニックが登場しています．1570〜1630 行に注目してください．これは，ゲームが始まってからビームを移動させるための，キー入力のために必要な部分なのです．

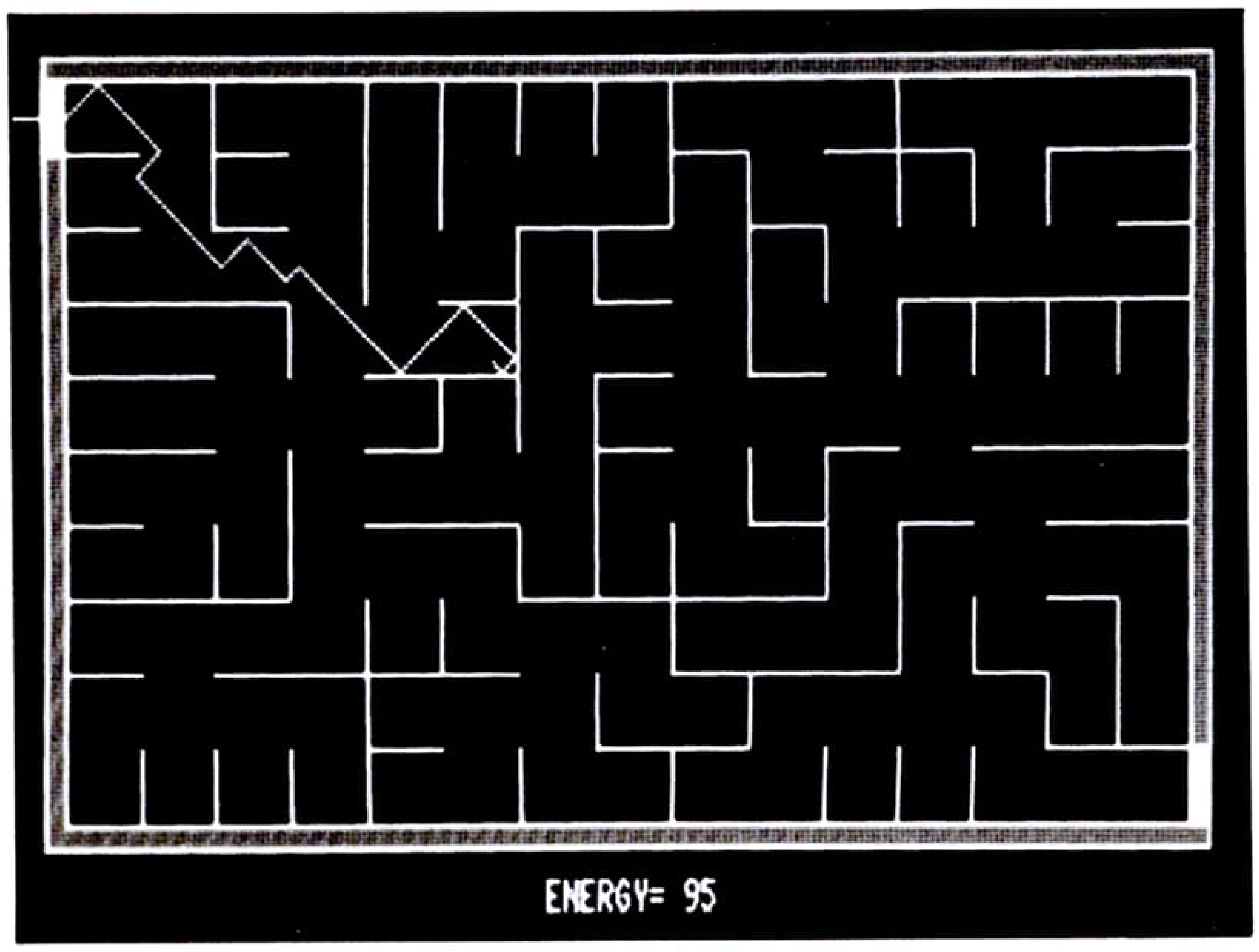
あるキーを入力して，その方向にビームを反射させる場合，

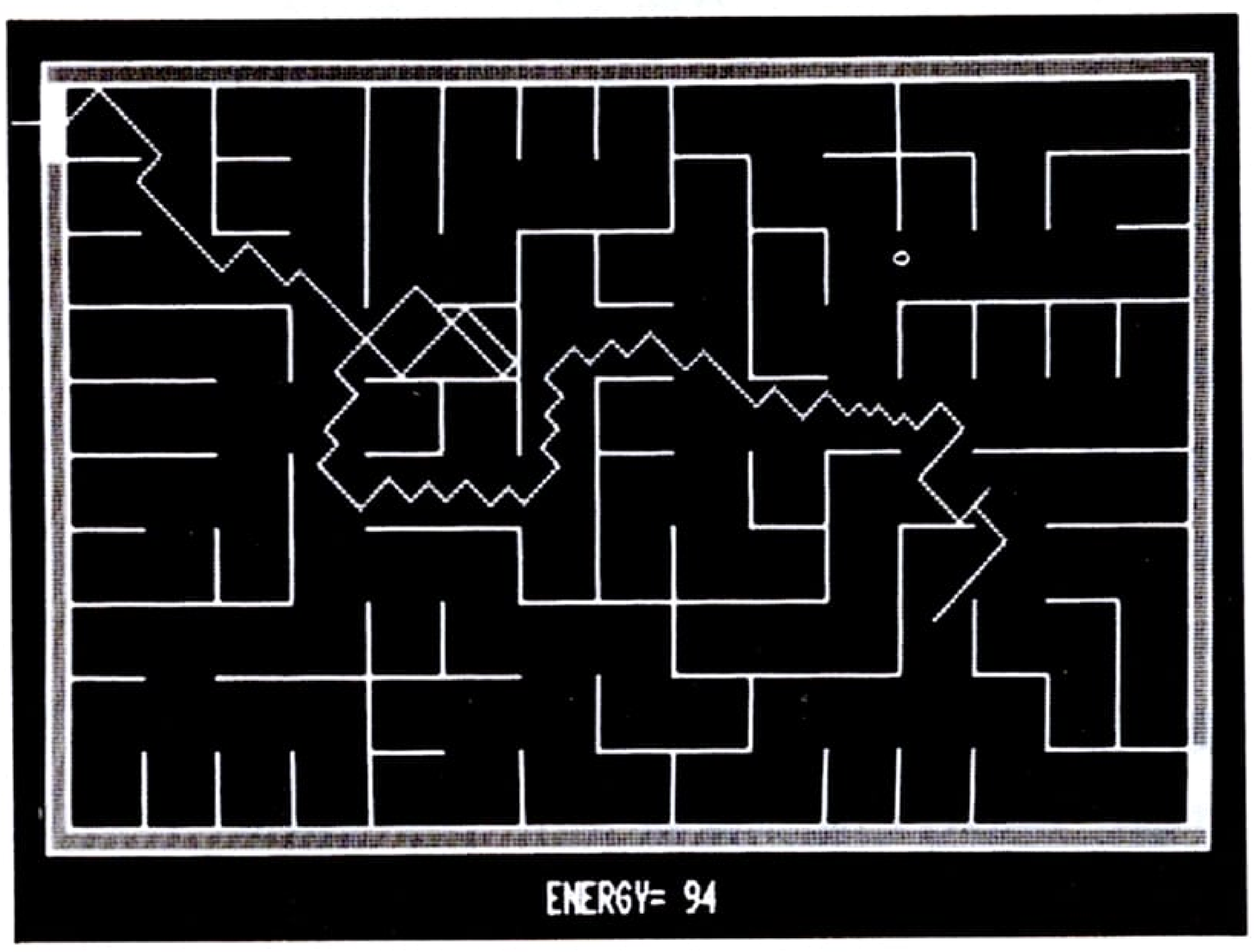
(1) キー入力を待つ
(2) キーを判断して動かす量を決定する
(3) その量の分だけビームを移動する

という 3 つの段階が考えられます．

キー入力は，INP 関数で行っています．テンキーの [0] 〜 [7] が押されているかどうかを INP(0) によって知ることができるのです．INP(0) の返す値としては，0〜255 までが考えられますから，あらかじめ配列変数 XS と YS を 256 個分用意し，キーが押されたとき，その配列変数 XS，YS からキーに対する移動方向を引き出せるように，データを入れておくのです．もちろん，必要のないキーが押されたとき INP(0) が返す値に対応する配列変数 XS，YS の値は，0 にします．







例えば，テンキーの[1]だけが押されているときINP(0)の値は253です．そこで，[1]が押されてから次のキーが押されるまではx軸方向に−1，y軸方向に0.5ずつ動かしたいのなら，XS(253)=−1，YS(253)=0.5とします．

[2]，[4]，[5]が押されたときも同様です．それ以外のキーが押されたときには，ビームの方向を変更してはいけません．そこで[1]，[2]，[4]，[5]以外のキーに対応するXS(n)，YS(n)の値は，すべて0とします．準備には多少手間がかかるものの，キー入力がなされた段階では，すばやく処理することが，可能となるのです．

以上の準備をしている部分が，1570～1630行なのです．

1650～1720行は，ビームがスタート位置まで動いていき，そこでドアが描かれるという，デモの部分です．

1740，1750行は，何かキーが押されたらゲームを開始するという，待ち時間の部分です．1750行の，

```
DUMMY$=INPUT$(1)
```

は，何か1文字キー入力があるまで，そこで実行を止めておくために利用しています．

● 1060～1210行

1060～1130行は，ビームを移動させていくためのメインの部分です．まず1080行でキー入力を確かめ，キーが押されていたら，それに対応する移動量を配列XS，YSから参照してきます．もしキー入力がない場合は，今までビームが移動してきた量XSと

YS を変更せずに，そのまま用います．

その値で，次にビームを表示するべき場所 XN，YN を決定し，1110 行でそこに壁があるかないかを判定します．そこに壁がない場合は，XD，YD にその座標を移して点を打ち，再びキー入力に戻ります．また，もし（XN，YN）に壁があるときには，実行は 1150 行に移ります．

1150 行では，ビームが出口の壁に当たったのかどうかを判断しています．出口の壁だった場合には，1810 行に飛んで，勝った場合の処理を行います．

1170，1180 行では，ビームの進む先を調べて，壁がなければすぐにメインの部分に戻り，横の壁があれば x 軸方向の移動量 XS の正負を逆に，縦の壁があれば y 軸方向の移動量 YS の正負を逆にします．

1190〜1210 行では壁に当たったということでエネルギーを示す変数 COUNT を1つ減らして表示し，BEEP 音を出します．もしエネルギーが 0 になったら，ゲームは負けで，1890 行以降の負けた場合の処理に実行が移ります．その他の場合は，再びメインの部分に戻ります．

- 1810〜1870 行

ゲームに勝った場合の処理をする部分です．

- 1890〜1910 行

ゲームに負けた場合の処理をする部分です．

キー入力のテクニック以外にはこれといって新しい方法は使っていませんが，なかなかおもしろいゲームになったでしょう．あなたも，他のプログラムを利用して，何かゲームを作ってみてください．始めは，そのプログラムに，ほんの少し手を加えるだけでよいのですから！

List 6.1

```
1000 ' List 6.1
1010 '
1020 WIDTH 80,25 : CONSOLE ,,0,1 : COLOR 7
1030 GOSUB *MAZE
1040 GOSUB *START : OUT 230,0
1050 '
1060 *MAIN
1070   IF INP(9)=127 THEN OUT 230,3 : END
1080   I=INP(0)
1090   IF I<255 THEN XS=X(I) : YS=Y(I)
1100   XN=XD+XS : YN=YD+YS
1110   IF POINT(XN,YN)=7 THEN GOSUB *CHECK
1120   XD=XN : YD=YN : PSET(XN,YN),5
1130 GOTO *MAIN
1140 '
```

（1060〜1130 行：キー入力によりビームを移動させるルーチン）

```
1150 *CHECK
1160 IF XN>=XGOAL AND YN>=YGOAL GOTO *WIN
1170 IF POINT(XN,YD)=7 THEN XS=-XS : XN=XN+2*XS
1180 IF POINT(XD,YN)=7 THEN YS=-YS : YN=YN+2*YS
1190 LOCATE 35,24  : PRINT USING "ENERGY=###";COUNT-1;
1200 COUNT=COUNT-1 : IF COUNT=0 GOTO *LOSE
1210 BEEP 1 : BEEP 0 : RETURN
1220 '
1230 *MAZE
1240 XMIN= 60 : YMIN= 20
1250 XMAX=580 : YMAX=180
1260 M=15     : N=10
1270 S=(M-1)*(N-1)
1280 XW=(XMAX-XMIN)/M
1290 YW=(YMAX-YMIN)/N
1300 XS(0)=-1 : XS(1)=1 : XS(2)=0 : XS(3)=0
1310 YS(0)=0  : YS(1)=0 : YS(2)=1 : YS(3)=-1
1320 RANDOMIZE VAL(RIGHT$(TIME$,2))
1330 '
1340 SCREEN 0,0 : CLS 3
1350 LINE  (XMIN-10,YMIN-5)-(XMAX+10,YMAX+5),7,B
1360 LINE  (XMIN,YMIN)-(XMAX,YMAX),7,B
1370 PAINT (XMIN-1,YMIN-1),CHR$(0)+CHR$(85)+CHR$(85),7
1380 LINE  (XMIN-10,YMIN+1)-(XMIN,YMIN+YW-1),0,BF
1390 LINE  (XMAX,YMAX-1)-(XMAX+10,YMAX-YW+1),0,BF
1400 '
1410 *LOOP : X=INT(RND*(M+1)) : Y=INT(RND*(N+1))
1420   XO=XMIN+X*XW : YO=YMIN+Y*YW
1430   IF POINT(XO,YO)=0 GOTO *LOOP
1440   FOR I=1 TO 8
1450     R=INT(RND*4)
1460     XN=XO+XS(R)*XW : YN=YO+YS(R)*YW
1470     IF XN<=XMIN OR XN=>XMAX GOTO 1520
1480     IF YN<=YMIN OR YN=>YMAX GOTO 1520
1490     IF POINT(XN,YN)<>0 GOTO 1520
1500     LINE (XO,YO)-(XN,YN),7
1510     XO=XN : YO=YN : S=S-1
1520   NEXT I
1530 IF S>0 GOTO *LOOP
1540 BEEP : RETURN
1550 '
```

壁にあたったかどうかの判定ルーチン（1160～1210）

迷路の大きさを設定（1240～1260）

迷路を作るサブルーチン（1230～1550）

```
1560 *START
1570 DIM X(255),Y(255)
1580 RESTORE 1630
1590 FOR I=1 TO 4
1600   READ A,X,Y
1610   X(A)=X : Y(A)=Y
1620 NEXT I
1630 DATA 253,-1,.5, 239,-1,-.5, 223,1,-.5, 251,1,.5
1640 '
1650 XD=XMIN+1 : YD=YMIN+YW/2 : XS=1 : YS=-.5
1660 FOR X=0 TO XMIN+1
1670   PSET (X,YD),5
1680   BEEP 1 : BEEP 0
1690   FOR T=1 TO 50 : NEXT T
1700 NEXT X
1710 BEEP 1 : LINE (XMIN-10,YMIN)-(XMIN,YMIN+YW),7,BF
1720 LINE (XMAX,YMAX-YW)-(XMAX+10,YMAX),7,BF : BEEP 0
1730 '
1740 LOCATE 28,24 : PRINT "Hit any key to start !!";
1750 DUMMY$=INPUT$(1) : CLS
1760 '
1770 COUNT=100 : XGOAL=XMAX : YGOAL=YMAX-YW
1780 LOCATE 35,24 : PRINT USING "ENERGY=###";COUNT;
1790 RETURN
1800 '
1810 *WIN
1820 FOR I=1 TO 3
1830   BEEP 1 : FOR T=1 TO 10 : NEXT T : BEEP 0
1840   FOR J=1 TO 100 : NEXT J
1850 NEXT I
1860 CLS : LOCATE 35,24 : PRINT "You win !!";
1870 LOCATE 0,0 : OUT 230,3 : END
1880 '
1890 *LOSE
1900 BEEP : CLS : LOCATE 35,24 : PRINT "You lose ...";
1910 LOCATE 0,0 : OUT 230,3 : END
```

キー入力に際して必要な初期設定

ゲーム開始時のデモンストレーション

ゲーム終了時の処理ルーチン

バイオリズム

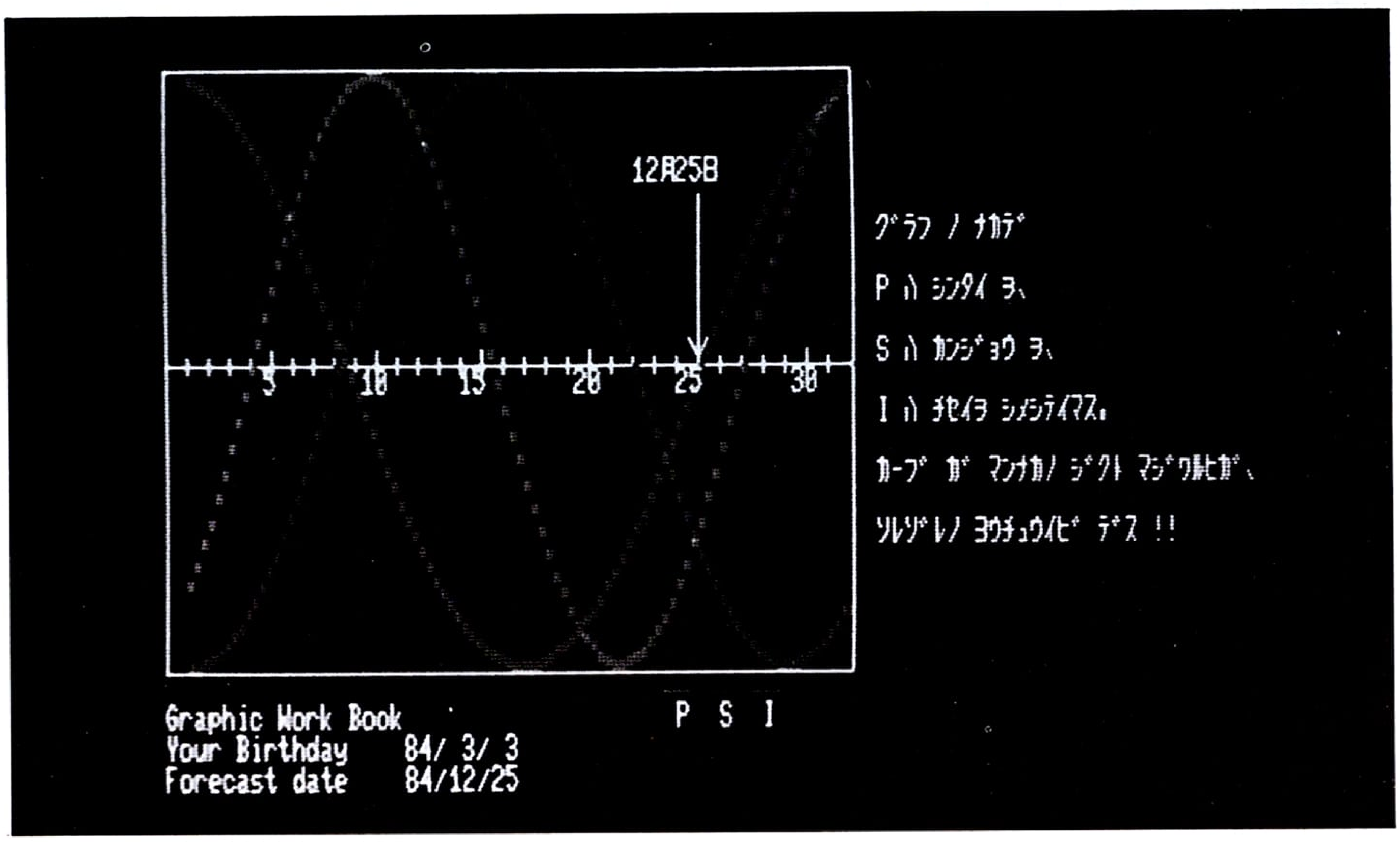
本書の中にも何度か登場した，sin，cos関数のカーブ，あれを見て，みなさんは何を連想するでしょうか？　海の波，それともヘビか何かが動いている様子でしょうか？実は，ゲームセンターなどでよく見かけるバイオリズムのあのカーブも，sin関数を使って描かれているものなのです．では，さっそく，そのバイオリズムのカーブを出力してみることにしましょう．

利用法

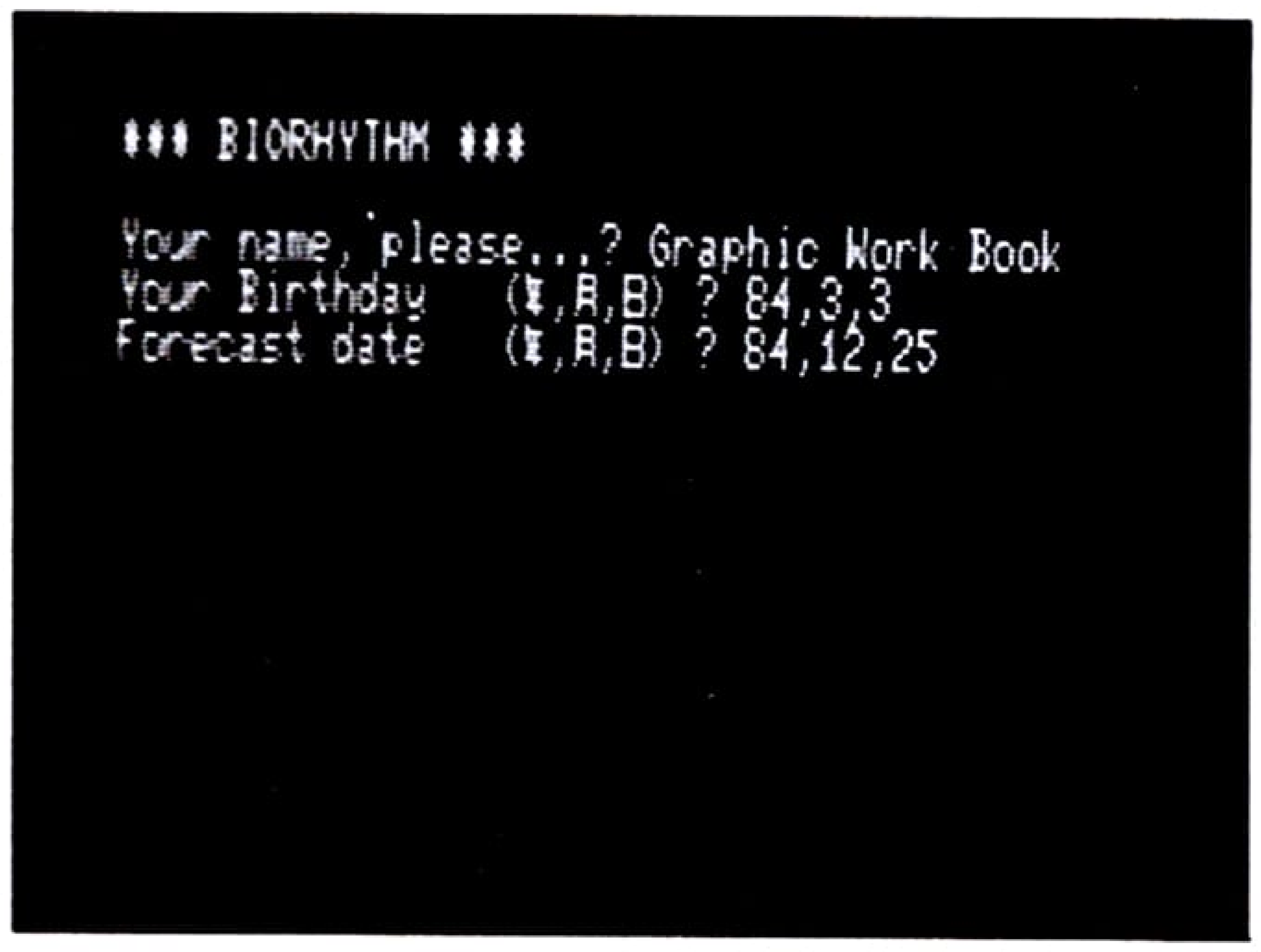
プログラムを実行すると，まず名前を聞いてきますので，入力してください．次に，あなたの誕生日を聞いてきますので，入力してください．このとき，年は必ず西暦で入力します．つまり昭和59年ならば1984年となるのです．最後に，予想したい日を入力します．

なお，存在しない日付を入力すると，再度入力をうながしてきます．

その後，身体（P），感情（S），知性（I）についての，それぞれの曲線が，予想日を含めた1か月分出力されます．横軸の目盛りは，1日の長さを示しています．さらに，予想日は，白い矢印で示されます．この矢



印が引かれる位置は，その予想日が始まる真夜中の0時になっていることに注意してください．



プログラムの説明

このバイオリズムのプログラムは，大きく分けて2つの部分から構成されています．今まで生きてきた日数を計算する部分（このとき，うるう年に注意！）と，計算したデータをもとに，サインカーブを描かせる部分（身体，感情，知性で，それぞれ，周期が異なる）です．

● 180～240行

名前，誕生日，予想日の入力部分です．日付の入力には，680行以降のサブルーチンを利用します．

● 680～870行

この部分は，日付を入力し，さらに西暦0年1月1日（紀元前1年1月1日）から，その日付までの日数を計算するサブルーチンです．

まず，1月～12月までの1か月あたりの日数を，それぞれ配列変数ML(n)に取り込みます．次に，750行で年，月，日を入力した後， 770， 780行でその年がうるう年かどうかを判断しています．うるう年は基本的には4で割り切れる年ですが，100で割れて400で割れない年はうるう年ではありません．770行の式は，うるう年ならば，F=1，そうでなければF=0という結果を与えます．従って，うるう年ならば，780行でML(2)が29日に変更されます．790，800行で，M月D日という日付が実際に存在するかどうかチェックします．

820行では，指定したY年の前年までのすべての日数を計算しています．さらに，830～850行で，M月の前の月までの日数を足し，最後に860行でDを足して合計の日数を求めています．

● 260～550行

ここでは，それぞれのグラフを出力します．まず270行で，身体PW，感情SW，知性IWの周期を決めます．280行では，誕生日から予想日の月始めまでの日数を計算し，変数DDに代入しています．このDDは，あなたが生まれてから，予想日の月始めまで生きてきた日数です．

290行は，身体，感情，知性の周期でDDを割り，その余りを求めることで，予想日の月始めが，それぞれの周期のどこに位置しているかを判定しています．

310～390行はグラフの枠を描いている部分です．

410～450行では，身体，感情，知性について，それぞれの周期をもとに，グラフを描いています．さらに，月始めにそれぞれの周期のどの位置にあるのかを計算した結

果であるP，S，Iと，日付を進めていく変数Tを足しています．

470～500行では，指定した予想日を矢印で示し，その日付を表示しています．

520～550行では，グラフの説明を表示しています．

● 570～660行

これは，日付を表示し，画面の右にバイオリズムの簡単な説明を出力している部分です．

バイオリズムの読み取り方としては「カーブが真ん中の軸と交差する日を要注意とします．この日は，事故などを起こしやすいので気をつける」というのが一般的です．詳しくは，巻末の参考文献をご覧ください．

List 6.2

```
100 ' List 6.2
110 '
120 WIDTH 80,25 : CONSOLE ,,0,1 : COLOR 7
130 SCREEN 0,0 : CLS 3
140 '
150 DIM ML(12)
160 DEF FNM(X,Y)=X-INT(X/Y)*Y
170 '
180 PRINT "*** BIORHYTHM ***" : PRINT
190 BEEP 1 : FOR T=1 TO 50 : NEXT T : BEEP 0
200 INPUT "Your name, please...";NA$
210 MS$="Your Birthday " : GOSUB *CALC
220 MB=M : DB=D : YB=Y : BL=L
230 MS$="Forecast date " : GOSUB *CALC
240 MF=M : DF=D : YF=Y : FL=L
250 '
260 CLS
270 PW=23 : SW=28 : IW=33
280 FL=FL-DF : DD=FL-BL
290 P=FNM(DD,PW) : S=FNM(DD,SW) : I=FNM(DD,IW)
300 '
310 LINE (0,0)-(386,160),7,B
320 LINE (0,79)-(386,79),7
330 FOR T=1 TO 31
340   IF T MOD 5=0 THEN YS=4 ELSE YS=2
350   LINE (T*12,79-YS)-(T*12,79+YS),7
360 NEXT T
370 FOR T=5 TO ML(MF) STEP 5
380   LOCATE (T*12-12)/8,10 : PRINT USING "##";T;
390 NEXT T
400 '
```

（180～240行）名前，誕生日，予想日の入力

（310～390行）グラフの枠を描く

```
410 FOR T=1 TO 32 STEP 1/3 : X=T*12
420   PD=79-78*SIN((T+P)/PW*6.283) : LINE (X,PD)-STEP(4,2),1,BF
430   SD=79-78*SIN((T+S)/SW*6.283) : LINE (X,SD)-STEP(4,2),2,BF
440   ID=79-78*SIN((T+I)/IW*6.283) : LINE (X,ID)-STEP(4,2),4,BF
450 NEXT T
```

グラフを描く

```
460 '
470 X=DF*12
480 LINE (X,34)-(X,77),7  : LINE -(X-5,72),7
490 LINE        -(X,77),7  : LINE -(X+5,72),7
500 LOCATE (X-DF*1.3)/8,3 : PRINT USING "##月##日";MF,DF;
```

予想日を矢印で示し，その日付を表示する

```
510 '
520 LINE (280,165)-(295,165),1
530 LINE (305,165)-(320,165),2
540 LINE (330,165)-(345,165),4
550 LOCATE 36,21 : PRINT "P  S  I"
```

グラフの説明

```
560 '
570 LOCATE 0,21  : PRINT NA$
580 PRINT USING "Your Birthday  ####/##/##";YB,MB,DB
590 PRINT USING "Forecast date  ####/##/##";YF,MF,DF
600 LOCATE 50,5  : PRINT "ｸﾞﾗﾌ ﾉ ﾅｶﾃﾞ"
610 LOCATE 50,7  : PRINT "P ﾊ ｼﾝﾀｲ ｦ、"
620 LOCATE 50,9  : PRINT "S ﾊ ｶﾝｼﾞｮｳ ｦ、"
630 LOCATE 50,11 : PRINT "I ﾊ ﾁｾｲｦ ｼﾒｼﾃｲﾏｽ。"
640 LOCATE 50,13 : PRINT "ｶｰﾌﾞ ｶﾞ ﾏﾝﾅｶﾉ ｼﾞｸﾄ ﾏｼﾞﾜﾙﾋｶﾞ、"
650 LOCATE 50,15 : PRINT "ｿﾚｿﾞﾚﾉ ﾖｳﾁｭｳｲﾋﾞ ﾃﾞｽ !!   ";
660 DUMMY$=INPUT$(1) : END
```

コメント

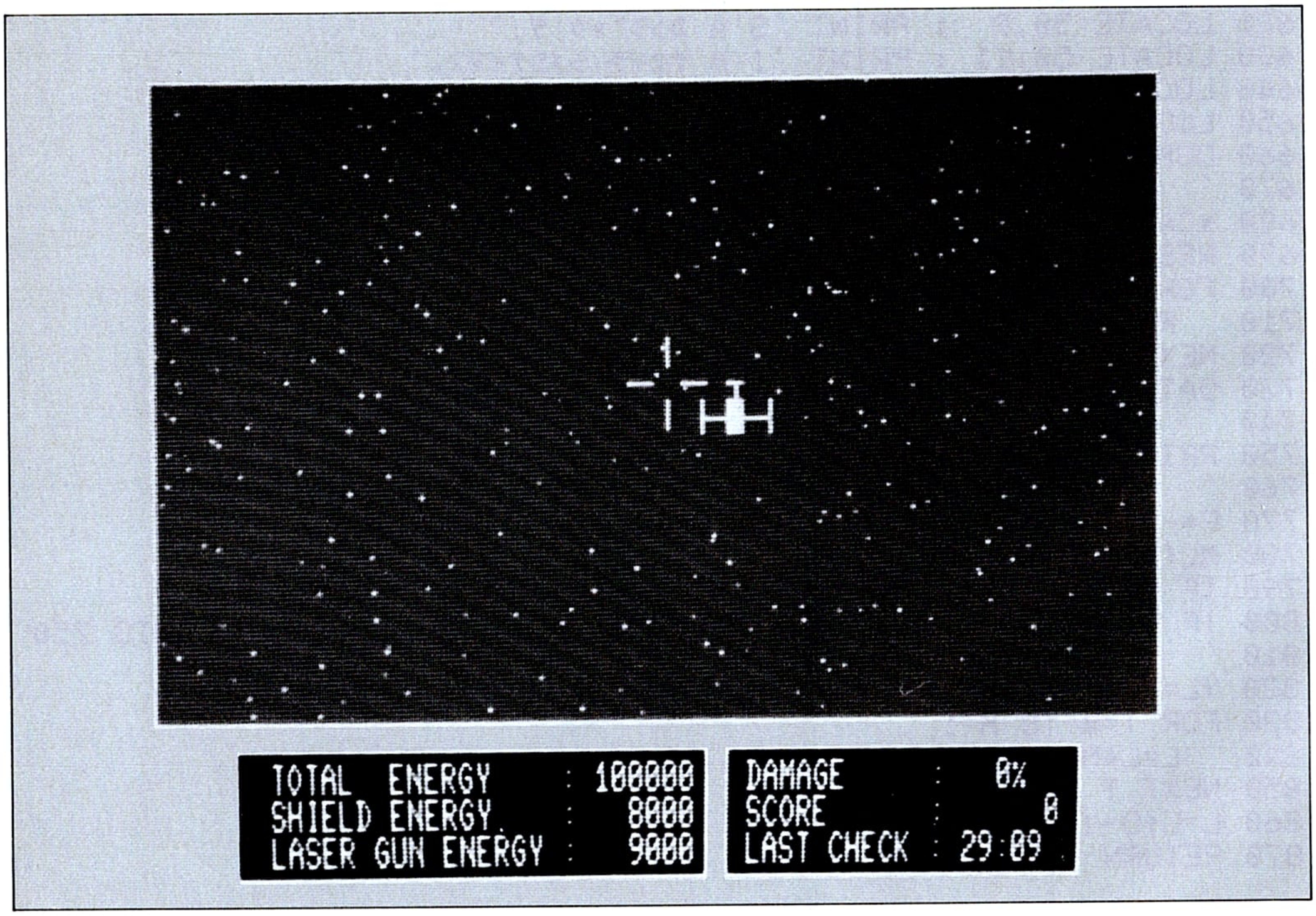
```
670 '
```

西暦0年1月1日からINPUTされた年月日までの日数を計算するルーチン

```
680 *CALC
690 RESTORE 730
700 FOR T=0 TO 12
710   READ ML(T)
720 NEXT T
730 DATA 0,31,28,31,30,31,30,31,31,30,31,30,31
740 '
750 PRINT MS$+"  (年,月,日) "; : INPUT Y,M,D
760 '
770 F=-((Y MOD 4)=0)+((Y MOD 100)=0)-((Y MOD 400)=0)
780 ML(2)=28+F
790 IF M<1 OR M>12 THEN PRINT M;"月 ﾊ ｱﾘﾏｾﾝ" : GOTO 750
800 IF D<1 OR D>ML(M) THEN PRINT M;"月";D;"日 ﾊ ｱﾘﾏｾﾝ" : GOTO 750
810 '
820 YL=Y : L=YL*365+INT(YL/4)-INT(YL/100)+INT(YL/400)
830 FOR T=0 TO M-1
840   L=L+ML(T)
850 NEXT T
860 L=L+D-1
870 RETURN
```

STAR WARSゲーム

4章で紹介したプログラムの中に，グラフィック・キャラクタを利用した宇宙船を動かすゲームがありました.「PC-8801 mkII ほどのグラフィック機能があるマシンで，どうしてグラフィック・キャラクタなどという稚拙なものを使わなくてはならんのか」と不思議に思った読者もおられるかと思います．しかし，コンピュータが持っている特有の機能を十分に活かしてプログラムを組んでいくということは，とても大切なことです．ゲームを作るにしてもその他のプログラムを作るにしても，そうしなければより楽しくよりおもしろいプログラムを組めるようになりません．というわけでここでは，グラフィック・キャラクタを，使った“STAR WARS ゲーム”を作ってみました．

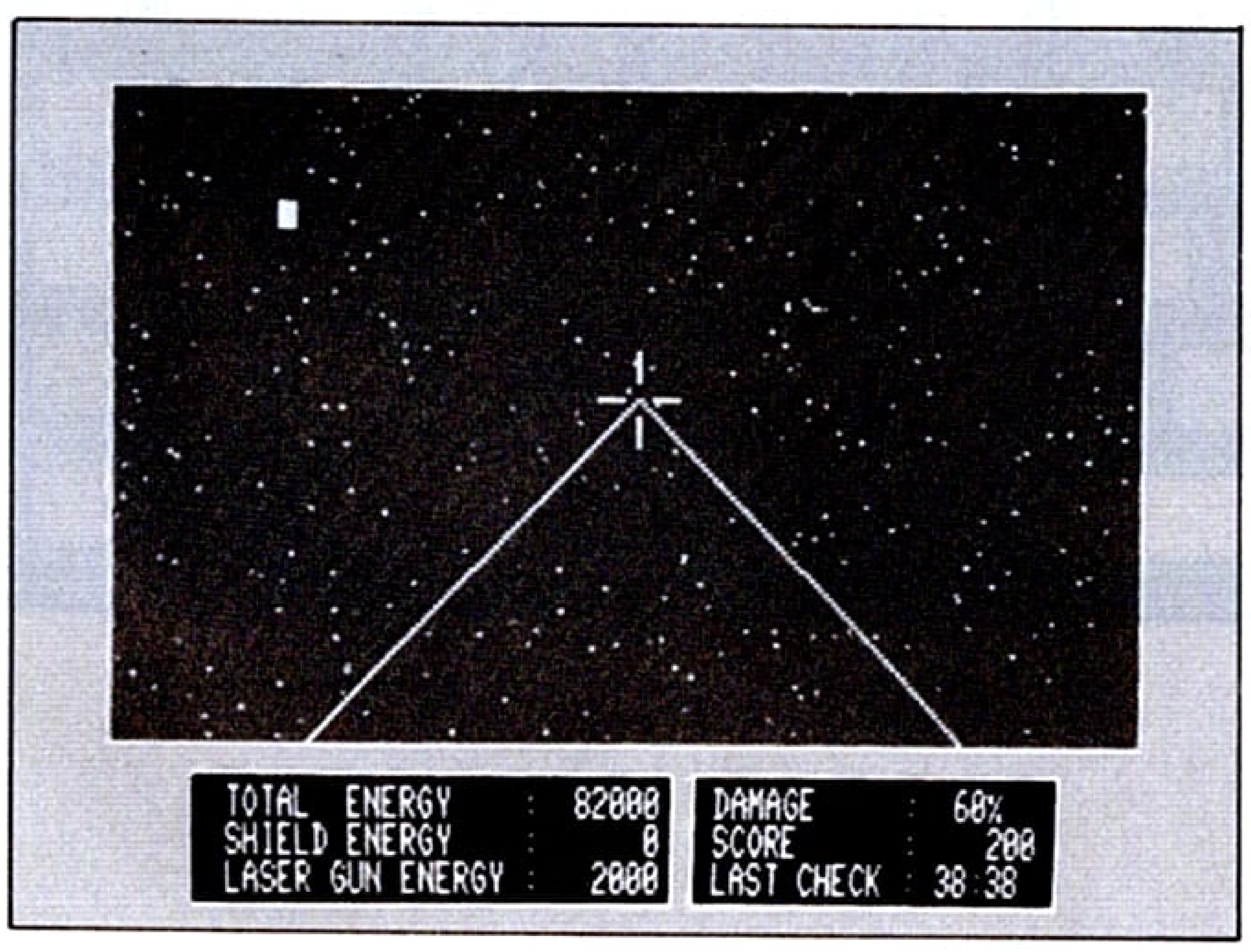


利用法

このゲームは，こちらに向かって迫ってくる未確認飛行物体(つまりUFO)をビーム砲で打ち落とすという，少し過激なゲームです．コックピットにいるあなたは，敵が発するビーム砲をかわしながら，攻撃用燃料を補給し，しかも敵を打ち倒さなければなりません．

画面の敵機を中央に寄せるのに 5 を中心としたテンキーを使います．例えばキー 8 を押せば敵機が上へ寄っていきます．画面中央の照準に敵機の中心をとらえたら，スペースキーを押して打ち落とします．ただし，ビーム砲は画面のLASER ENERGYが1000以上ないと発射できません．エネルギーが足りない場合はカーソルキー → で補充してください．カーソルキー ← では逆にエネルギーが減ります．

また敵機がBEEP音を発して赤く光るときは，キー 5 を押してシールドをはり，防御しないと敵のビーム砲に打たれたことになり，DAMAGEが増加します．シールドをはらなかったときどうなるのかは，見てからのお楽しみです．シールドをはるごとにSHIELD ENERGYが1000ずつ減るので，カーソルキー ↑ で補充します．カーソルキー ↓ を押すと逆にエネルギーが減ります．もしSHIELD ENERGYが0のとき攻撃されると，シールドがはれないばかりかDAMAGEが増えます．このDAMAGEが100％を超えるかTOTAL ENERGYが0になると，ゲームは終了します．ESC キーによってもゲームは終了します．



プログラムの説明

このプログラムで描いているUFOのキャラクタは，4章の〝グラフィック・キャラクタで宇宙船を！〟のテクニックを使っています．キー入力の方法など，その他の部分でも各章で説明したことを応用してプログラムを組んでいます．難しくてちょっと理解できないなと思った方は，前の部分を読み返してみてください．

- 1030行

  乱数系列を設定します．

- 1040～1070行

  キー入力に対する移動量を決めておきます．

- 1090～1220行

  UFOの形を，グラフィック・キャラクタとカーソルキーのキャラクタコードで定義します．近づいてくるにつれてUFOは3回形を変えます．最も近くに来たときのUFOの形も3種類あります．これはリスト1110行，1160行，1210行で決めています．

● 1240，1250 行

キー入力のための文字列を定義します.

● 1300～1400 行

宇宙船のコックピットをグラフィック画面に描きます.

● 1420～1440 行

点数，エネルギーの初期化と，表示をします.

● 1460～1490 行

何かキーを押すとゲームが始まります.

● 1550～1630 行

1530～1980 行が，このプログラムの中心となるループを構成している部分です.

まず 1560 行でどのキーが押されているか調べ，1570～1600 行でそのキーに対応する処理を行っています. 1610～1630 行では，同様にして，テンキーの入力を調べています. キー 5 が押された場合は，SFLAG を 1 に設定しています.

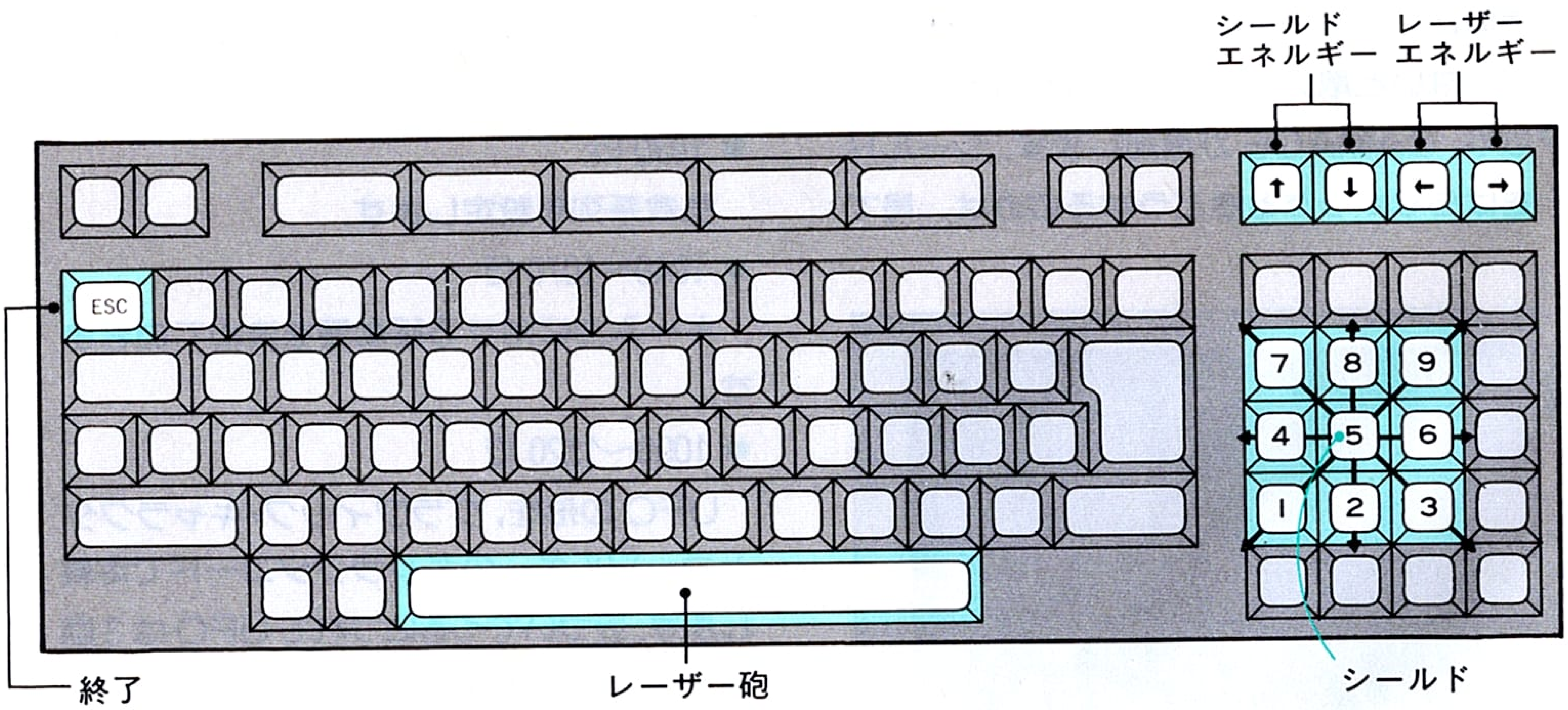
● 1650～1680 行

ここでいくつかのチエックをしています. シールドをはるときは 1660 行，敵の UFO が画面上にあるときは 1670 行，敵の UFO を発生させるかどうかは 1680 行でチエックして，それぞれのサブルーチンへ飛びます.

● 1760～1860 行

画面上の UFO を動かします.

まず 1770 行で XS，YS に−1，0，1 のうちどれかの数を発生させ，その値にキー入力による移動方向 XS(KY)，YS(KY) を加えた上で，UFO の距離 DISO を考えて，新しい UFO の位置 XN と YN を決めます. UFO が遠くにあるとき，つまり DISO が 0 に近い場合は，UFO の移動量が少なくなります. 距離を表す変数 DISO と DISN は 0～2 の値をとります. この値によって UFO の形が決まります. 次に 1790 行でその UFO がループの中で何回動かされたか



をカウントし，1800行でその数によってUFOの距離DISNを決めます．1810行で移動する前の位置にあるUFOを消し，1820〜1830行で移動可能かどうかを判断し，1840行で新しいUFOを描きます．UFOを動かす方法は，4章の“テニスゲームの仕組み”のボールを動かすテクニックと同じ方法です．

- 1880〜1910行

画面上のUFOが攻撃するかどうか決めます．1890行では，UFOが攻撃したにもかかわらずシールドをはらなかったときに，DAMAGEサブルーチンへ飛ぶ処理をしています．このDAMAGEサブルーチンは，エネルギーを減らすなどの処理をしています．

- 1930〜1960行

UFOの攻撃を画面に表示します．

- 2000〜2160行

画面にエネルギーや点数を表示します．

- 2180〜2310行

UFOの攻撃の際，シールドをはって防御できなかったときの処理をします．

- 2330〜2390行

シールドをはる処理をします．

- 2410〜2630行

敵のUFOを打ち落とす処理です．2490行で，命中したかどうか判断します．2580〜2620行でエネルギーと点数を増加させます．

2650〜2710行

SHIELD ENERGYの増減を行います．

- 2730〜2790行

LASER ENERGYの増減を行います．

- 2810〜2880行

終了時の処理です．

どうです？ 市販のゲームソフトにはかなわないけれど，手作りのゲームという感じで結構楽しめるでしょう．難しいテクニックを使わなくても，これ位のゲームは作れるのです．制作意欲がわいてきた方は，グラフィック画面のみを使って，このゲームを作り直してみるとよいでしょう．その場合，UFOはGET，PUT命令で動かすと簡単です．4章の“NIGHT FLIGHT”を参考にするとよいでしょう．よりよいゲームができるかどうかはあなた次第です．

List 6.3

```
1000 '
1010 ' List 6.3
1020 '
1030 RANDOMIZE VAL(RIGHT$(TIME$,2))……乱数系列の設定
1040 XS(0)=0  : XS(1)=1  : XS(2)=1  : XS(3)=0
1050 XS(4)=-1 : XS(5)=-1 : XS(6)=-1 : XS(7)=0  : XS(8)=1    定数の
1060 YS(0)=0  : YS(1)=0  : YS(2)=1  : YS(3)=1               設定
1070 YS(4)=1  : YS(5)=0  : YS(6)=-1 : YS(7)=-1 : YS(8)=-1
```

```
1080 '
1090 F$(0,0)="·" : F$(0,1)="■"
1100 D$(0,0)=" " : D$(0,1)=" "
1110 F$(0,2)=CHR$(30)+"┬"+STRING$(3,CHR$(29))+CHR$(31)+"├─■─┤"
1120 D$(0,2)=CHR$(30)+" "+STRING$(3,CHR$(29))+CHR$(31)+"     "
1130 '
1140 F$(1,0)="·" : F$(1,1)="■"
1150 D$(1,0)=" " : D$(1,1)=" "
1160 F$(1,2)=CHR$(30)+"┬"+STRING$(3,CHR$(29))+CHR$(31)+"┌┬■┬┐"
1170 D$(1,2)=CHR$(30)+" "+STRING$(3,CHR$(29))+CHR$(31)+"     "
1180 '
1190 F$(2,0)="·" : F$(2,1)="▬"
1200 D$(2,0)=" " : D$(2,1)=" "
1210 F$(2,2)=CHR$(29)+"├●┤"
1220 D$(2,2)=CHR$(29)+"   "
1230 '
1240 KY1$=CHR$(191)+CHR$(247)+CHR$(251)+CHR$(253)+CHR$(239)
1250 KY1$=KY1$+CHR$(127)+CHR$(223):KY2$=CHR$(254)+CHR$(253)
1260 '
1270 WIDTH 80,25 : CONSOLE ,,0,1 : COLOR 7
1280 SCREEN 0,0  : CLS 2
1290 '
1300 LINE (80,16)-(560,159),7,B
1310 LINE (119,165)-(340,193),7,B
1320 LINE (350,165)-(520,193),7,B
1330 PAINT (0,0),CHR$(85)+CHR$(85)+CHR$(85),7
1340 LINE (81,17)-(559,158),1,BF
1350 FOR I=1 TO 300
1360   X=RND*478+81:Y=RND*141+17 : C=INT(RND*3)+5
1370   PSET (X,Y),C
1380 NEXT I
1390 LINE (323,74)-(323,80),7 : LINE (323,88)-(323,94),7
1400 LINE (305,84)-(316,84),7 : LINE (330,84)-(341,84),7
1410 '
1420 *INIT
1430 TE=100000! : SE=10000 : LE=10000 : DM=0 : SC=0
1440 GOSUB *TEXT1 : GOSUB *TEXT2
1450 '
1460 *START
1470 LOCATE 28,18 : PRINT "Hit any key to start...";
1480 I$=INPUT$(1) : LOCATE 28,18 : PRINT SPACE$(25)
1490 FLAG=0 : AFLAG=0 : SFLAG=0
1500 '
1510 '
1520 '
```

グラフィック・キャラクタを決める（1080〜1220）

キー入力のための文字列を定義する（1240〜1250）

コックピットを描く（1270〜1400）

点数，エネルギーの初期化（1420〜1440）

ゲームスタート時の処理（1460〜1490）

```
1530 *MAINLOOP
1540 '
```

キー入力処理ルーチン

```
1550 *KEYINPUT
1560 P0=INP(0) : P1=INP(1) : P8=INP(8) : P9=INP(9) : PA=INP(10)
1570 IF P9=191 THEN GOSUB *FIRE : GOTO *MAINLOOP
1580 IF P8=253 OR PA=253 THEN GOSUB *SHIELD
1590 IF P8=251 OR PA=251 THEN GOSUB *LASER
1600 IF P9=127 THEN END
1610 KY=INSTR(KY2$,CHR$(P1))+6 : IF KY<>6 GOTO *CHECK
1620 KY=INSTR(KY1$,CHR$(P0))
1630 IF KY=7 THEN SFLAG=1 ELSE SFLAG=0
1640 '
1650 *CHECK
1660 IF SFLAG=1 THEN GOSUB *SHIELDON
1670 IF FLAG=1 GOTO *MOVE
1680 IF RND>.1 AND FLAG=0 GOTO *MAINLOOP
1690 '
1700 *GENERATE
1710 XO=INT(RND*56)+12 : YO=INT(RND*16)+3
1720 N=INT(RND*3)
1730 COUNT=0 : DISO=0 : FLAG=1
1740 BEEP 1 : BEEP 0
1750 '
```

UFOを動かす

```
1760 *MOVE
1770 XS=1-INT(RND*3) : YS=1-INT(RND*3)
1780 XN=XO+(XS+XS(KY))/(4-DISO) : YN=YO+(YS+YS(KY))/(4-DISO)
1790 COUNT=COUNT+1
1800 DISN=-(COUNT>=30 AND COUNT<80)-(COUNT>=80)*2
1810 LOCATE INT(XO),INT(YO) : PRINT D$(N,DISO)
1820 IF XN<12 OR XN>67 THEN FLAG=0 : AFLAG=0 : GOTO *MAINLOOP
1830 IF YN<3  OR YN>18 THEN FLAG=0 : AFLAG=0 : GOTO *MAINLOOP
1840 LOCATE INT(XN),INT(YN): PRINT F$(N,DISN)
1850 XO=XN : YO=YN : DISO=DISN
1860 BEEP 1 : BEEP 0
1870 '
```

UFOのビーム砲攻撃ルーチン

```
1880 *ATTACK
1890 IF AFLAG=1 AND SFLAG=0 THEN GOSUB *DAMAGE : GOTO *MOVE
1900 AFLAG=0
1910 IF RND>.05 GOTO *MAINLOOP
1920 '
1930 LOCATE INT(XN),INT(YN) : COLOR 3 : PRINT F$(N,DISN)
1940 BEEP 1 : FOR T=1 TO 90 : NEXT T : BEEP 0
1950 LOCATE INT(XN),INT(YN) : COLOR 7 : PRINT F$(N,DISN)
1960 AFLAG=1
1970 '
1980 GOTO *MAINLOOP
1990 '
```

```
2000 *TEXT1
2010 LOCATE 17,21 : PRINT "TOTAL  ENERGY    :"
2020 LOCATE 17,22 : PRINT "SHIELD ENERGY    :"
2030 LOCATE 17,23 : PRINT "LASER GUN ENERGY :"
2040 LOCATE 45,21 : PRINT "DAMAGE      :      %"
2050 LOCATE 45,22 : PRINT "SCORE       :"
2060 LOCATE 45,23 : PRINT "LAST CHECK :"
2070 RETURN
2080 '
2090 *TEXT2
2100 LOCATE 36,21 : PRINT USING "######";TE
2110 LOCATE 36,22 : PRINT USING "######";SE
2120 LOCATE 36,23 : PRINT USING "######";LE
2130 LOCATE 58,21 : PRINT USING "###";DM
2140 LOCATE 58,22 : PRINT USING "######";SC
2150 LOCATE 58,23 : PRINT RIGHT$(TIME$,5)
2160 RETURN
2170 '
2180 *DAMAGE
2190 AFLAG=0
2200 SE=SE-1000*(DISN+1)
2210 IF SE<0 THEN DM=DM+INT(-SE/100) : SE=0
2220 IF DM>=100 GOTO *FINISH
2230 '
2240 COLOR=(7,3)
2250 FOR T=1 TO 3
2260   WIDTH 80,20 : WIDTH 80,25
2270   BEEP 1 : BEEP 0
2280 NEXT T
2290 COLOR=(7,7) : WIDTH 80,25
2300 GOSUB *TEXT1 : GOSUB *TEXT2
2310 RETURN
2320 '
2330 *SHIELDON
2340 IF SE-1000<0 THEN RETURN ELSE SE=SE-1000
2350 COLOR=(1,7)
2360 BEEP 1 : FOR T=1 TO 10 : NEXT T : BEEP 0
2370 COLOR=(1,1)
2380 GOSUB *TEXT2
2390 RETURN
2400 '
2410 *FIRE
2420 IF LE<1000 THEN RETURN
```

画面にエネルギーや点数を表示する

UFOのビーム砲がシールドで防御できなかったときの処理

シールドをはる処理

```
2430 '
2440 LE=LE-1000
2450 BEEP 1
2460 LINE (323,84)-(173,158),6 : LINE (323,84)-(473,158),6
2470 LINE (323,84)-(173,158),1 : LINE (323,84)-(473,158),1
2480 BEEP 0
2490 IF INT(XN)<>40 OR INT(YN)<>10 THEN GOSUB *TEXT2 : RETURN
2500 '
2510 COLOR 2 :LOCATE INT(XN),INT(YN) :PRINT F$(N,DISN) :COLOR 7
2520 FOR T1=1 TO 5
2530   BEEP 1 : FOR T2=1 TO 8 : NEXT T2 : BEEP 0
2540   FOR T2=1 TO T1*12 : NEXT T2
2550 NEXT T1
2560 LOCATE INT(XN),INT(YN) : PRINT D$(N,DISN)
2570 '
2580 TE=TE+2000*(3-DISN) : SC=SC+100*(3-DISN)
2590 DM=DM-10*(3-DISN)  : IF DM<0 THEN DM=0
2600 SE=SE+3000
2610 FLAG=0 : AFLAG=0 : XN=0 : YN=0
2620 GOSUB *TEXT2
2630 RETURN
2640 '
2650 *SHIELD
2660 IF P8=253 THEN SES=1000 ELSE SES=-1000
2670 IF SE+SES<0 OR TE-SES<0 THEN GOSUB *TEXT2 : RETURN
2680 SE=SE+SES : TE=TE-SES
2690 BEEP 1 : BEEP 0
2700 GOSUB *TEXT2
2710 RETURN
2720 '
2730 *LASER
2740 IF P8=251 THEN LES=1000 ELSE LES=-1000
2750 IF LE+LES<0 OR TE-LES<0 THEN GOSUB *TEXT2 : RETURN
2760 LE=LE+LES : TE=TE-LES
2770 BEEP 1 : BEEP 0
2780 GOSUB *TEXT2
2790 RETURN
2800 '
2810 *FINISH
2820 GOSUB *TEXT2
2830 FOR T1=1 TO 5
2840   BEEP 1 : FOR T2=1 TO 5 :  NEXT T2 : BEEP 0
2850   FOR T2=1 TO 50 : NEXT T2
2860 NEXT T1
2870 *KEYCLEAR : IF INKEY$<>"" GOTO *KEYCLEAR
2880 LOCATE 0,0
```

UFOを打ち落とす処理

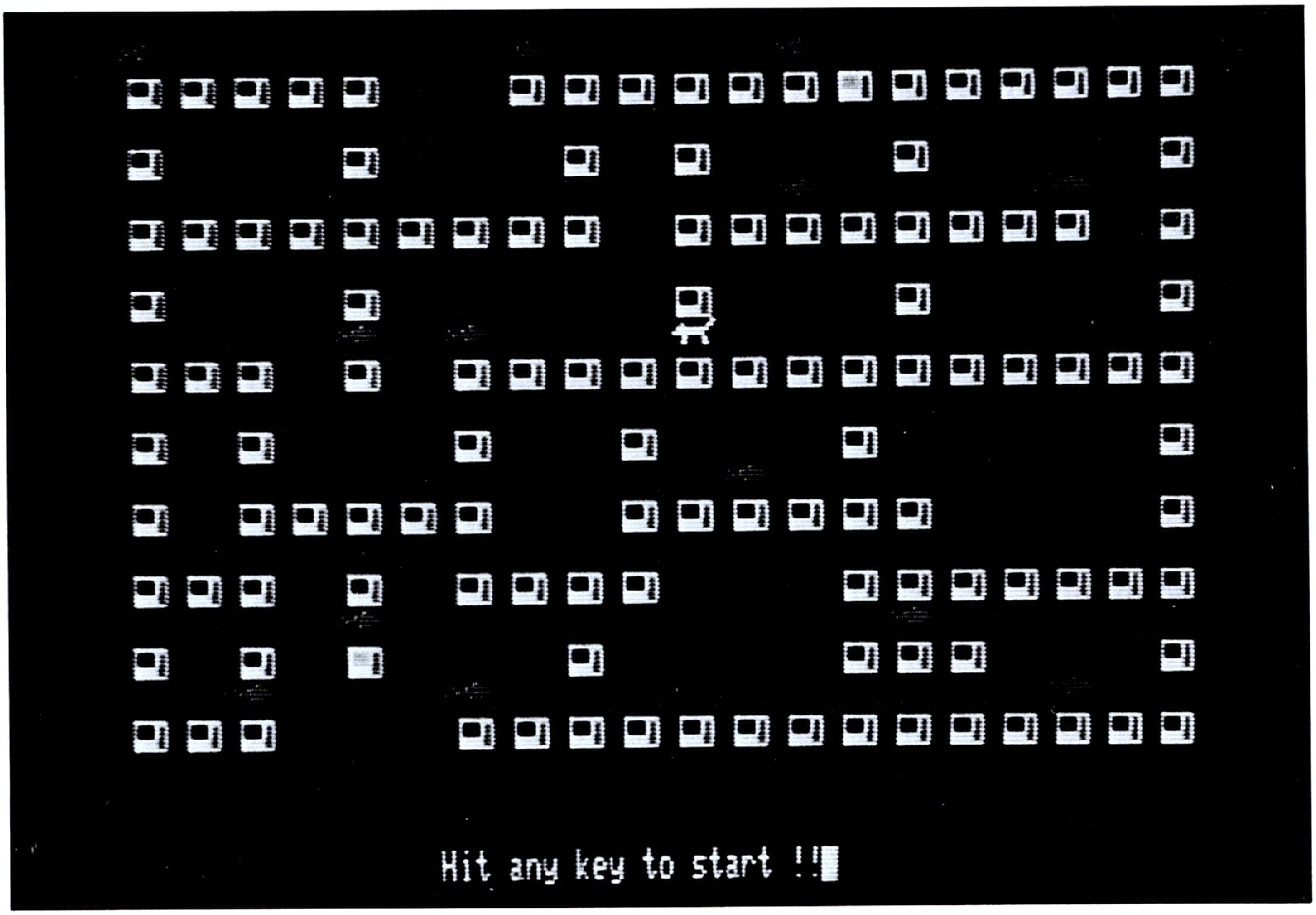
焼けてない トタン屋根の上のネコ

最後に，「ネコが，屋根から屋根へと飛び移り魚を取って行く」というゲームを考えてみました．それだけではあまりおもしろくありませんので「ネコの足音に気づいた人が，次々と部屋の明かりをつけてネコを追ってくる」ことにします．あなた，つまりネコは，その追手を振り切って，すべての魚を取らなければなりません．

このゲームは今までに出てきたテクニックを応用して作られています．じっくりとリストを眺めてゲームの作り方を研究してください．

利用法

魚を1匹取ると100点，1度捕まると，点



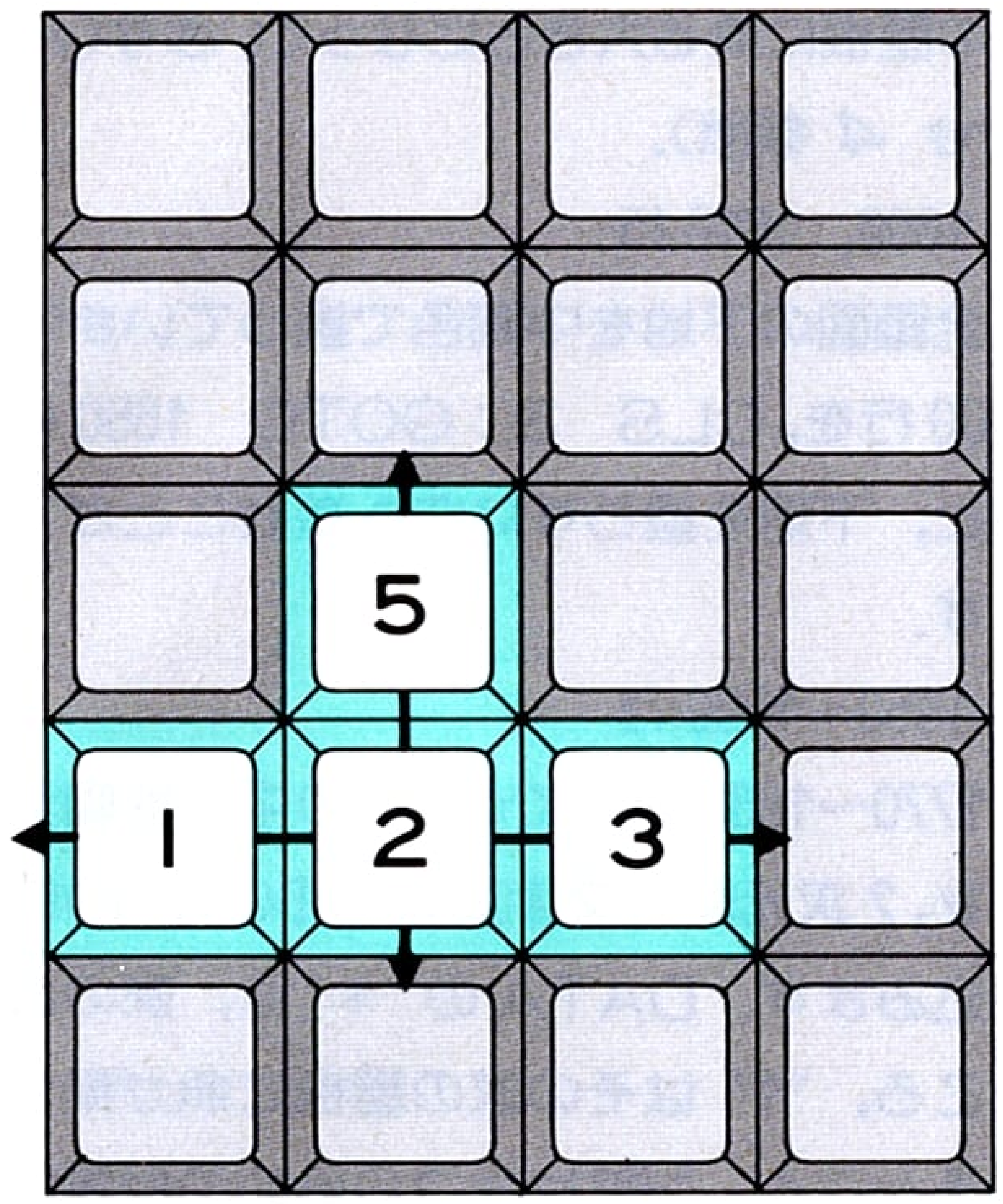


Figure 1

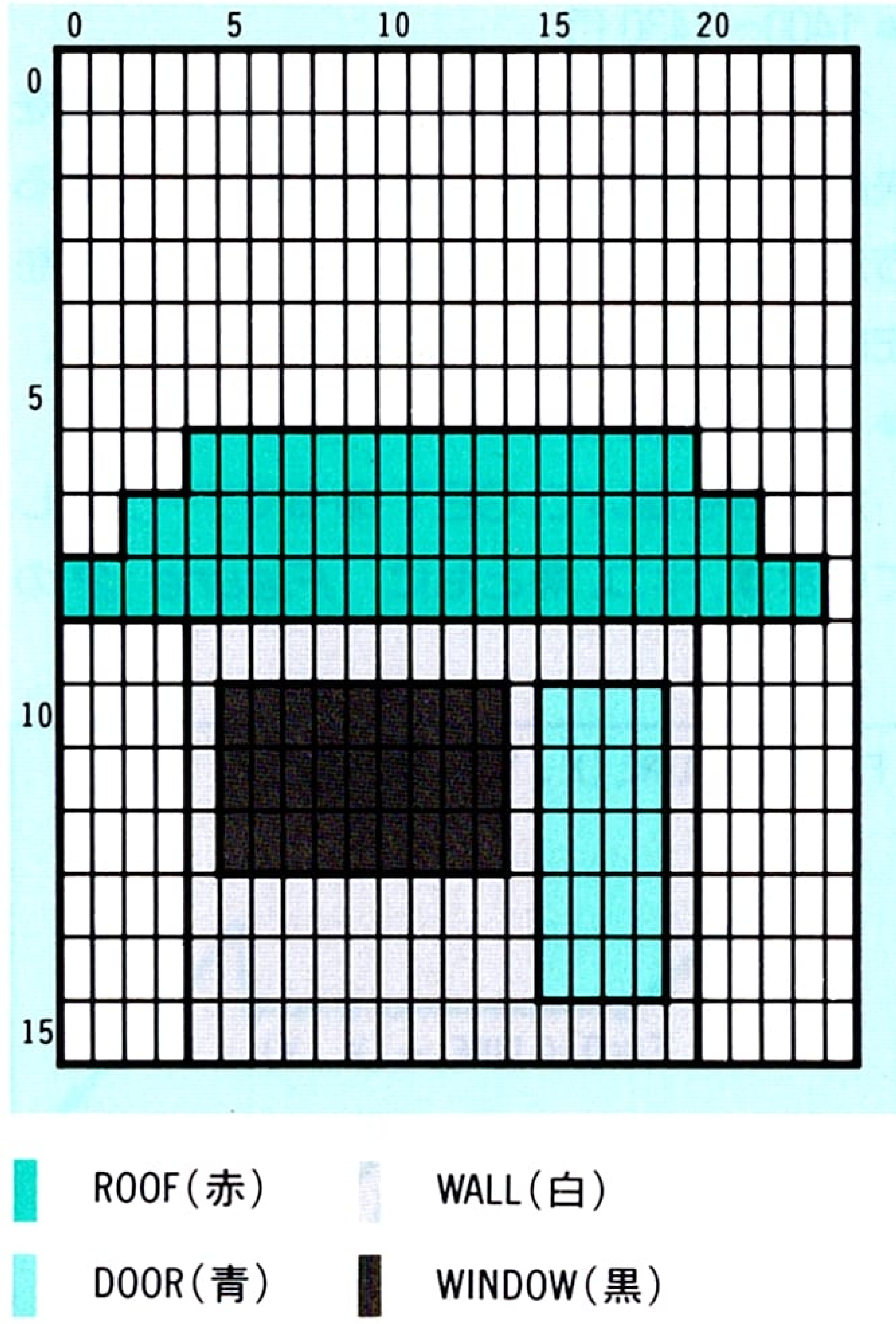


Figure 2

数は0に戻ります．ネコを操作するには，テンキーを用いますが，これは上下左右それぞれ5，2，1，3に対応しています．全部の魚を取ると，点数が出力されます（*Figure 1* 参照）．

プログラムの説明

このプログラムは，ブロックごとに書かれているので，上から順に理解していってください．

- 1020～1140 行

変数の定義，初期化をする部分です．

- 1180～1200 行

ゲームのキャラクタを GET 命令でデータに取り込むときに必要な部分の下地を塗っています．変数 BC1 と BC2 は混ぜ合わせるパレットコードで，結果は，BC1 と BC2 で表される色の中間色になります．これは，x 軸方向に BC1 と BC2 の色を交互に線を引いているためです．

- 1240～1290 行

LINE 命令で，グラフィックを描いて，GET 命令で，データを配列変数の HOUSE に取り込みます（*Figure 2* 参照）．

- 1330～1360 行

ここでは，右向きのネコを描いています．1450,1460行のDATAを読んで，LINE 命令を使いネコを描きますが，LINE −(X，Y)の形式を利用しています．LINE −(X，Y)が実行されて，次々と直線がつながっていき，ネコの形になります（*Figure 3* 参照）．

● 1400〜1430 行

左向きのネコを描きます．2 章の「位置を決める」で説明した，図形を裏返しにする方法を利用して，右向きのネコのデータを左向きのネコのデータに変換しています．

● 1500〜1560 行

魚の形を描いて，GET 命令でデータにしています．ネコ，魚ともに，*Figure 2* の家の屋根に乗る大きさになっています（*Figure 4* 参照）．

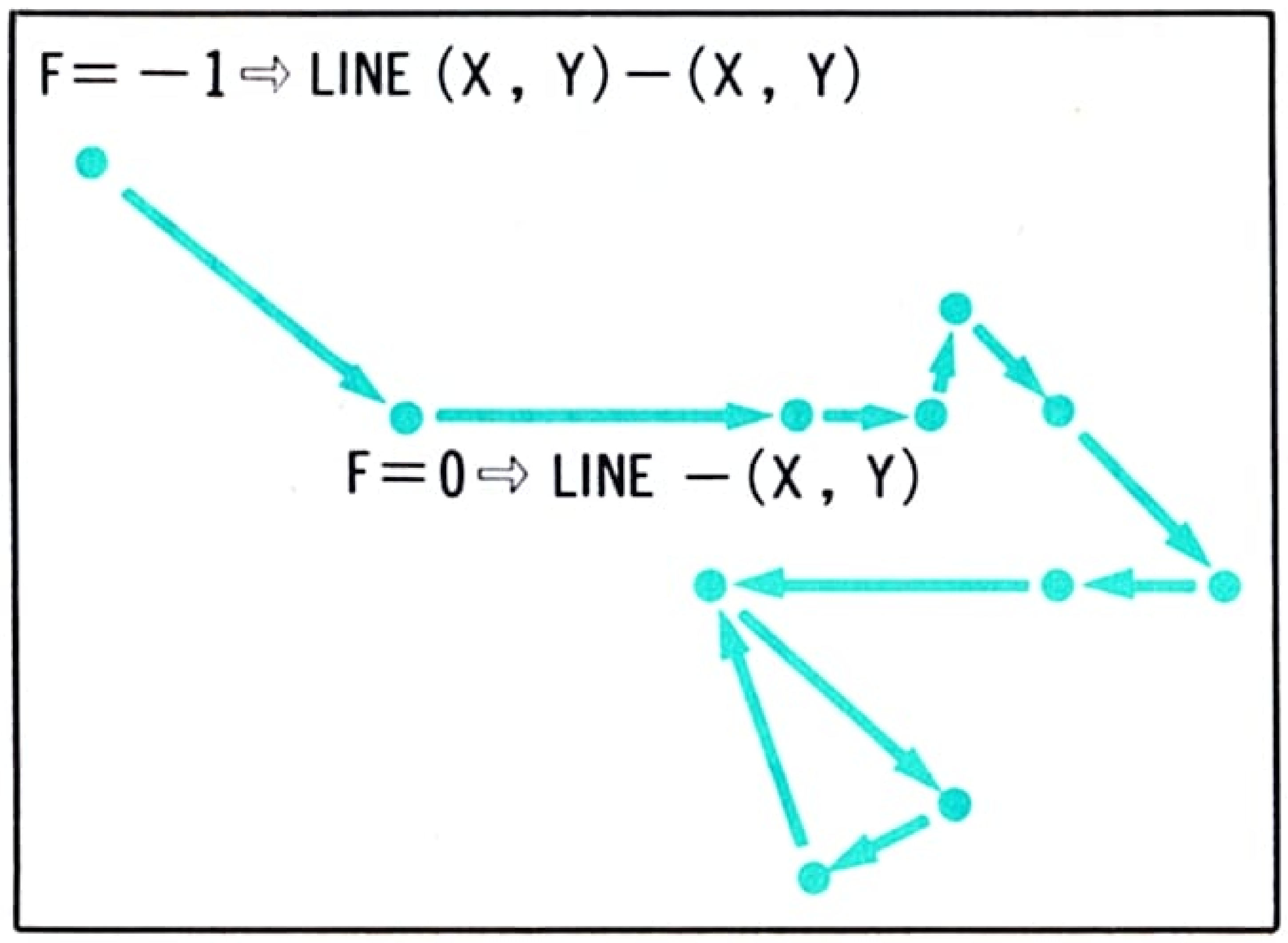


Figure 3

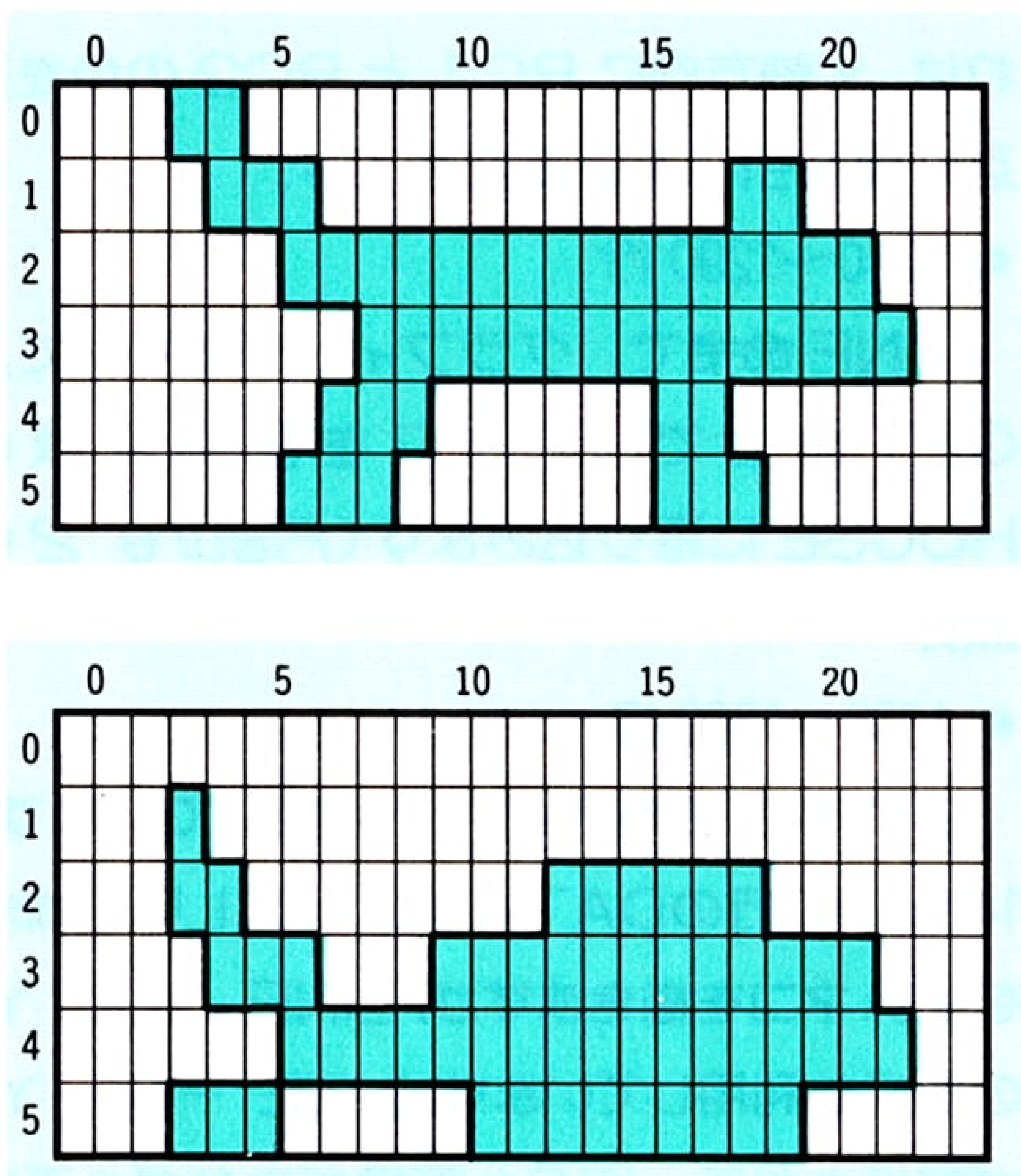


Figure 4

● 1600，1610 行

全画面の下地を中間色で塗っています．1600 行を，CLS　3：GOTO　1650 にすると，下地を塗らずに次へ進むことができます．

● 1650〜1860 行

1770〜1860 行までの，町の中の地図データを，2 次元配列文字変数 D$(X，Y)に取り込みます．DATA の“*”は，家のあるところ，“1”はその家の屋根に魚が乗っているところ，“@”はネコの始めの位置を示しています（*Figure 5* 参照）．ここで，魚の個数を FMAX と PC で数えています．この“*”，“1”，“@”は，横 20×縦 10 のマス目の中であれば，どこへ動かしてもかまいません．いろいろと変えてみてください．

● 1930〜2000 行

D$（X，Y）の変数を参照して，画面に家と魚を出力します．x 軸方向 0〜19 を画面の座標に変換する式は，GX=X * 26+50，y 軸方向 0〜9 に対応する GY は，GY=Y * 16+10 です．さらに 1980 行で魚を出力

		x						
		0	1	2	3	4	....	
	0	1	*	*	*	·	·	·
	1	*			*			
	2	*	*	*	@			
y	3	*			*			
	⋮	·						
	⋮	·						
	⋮	·						

Figure 5

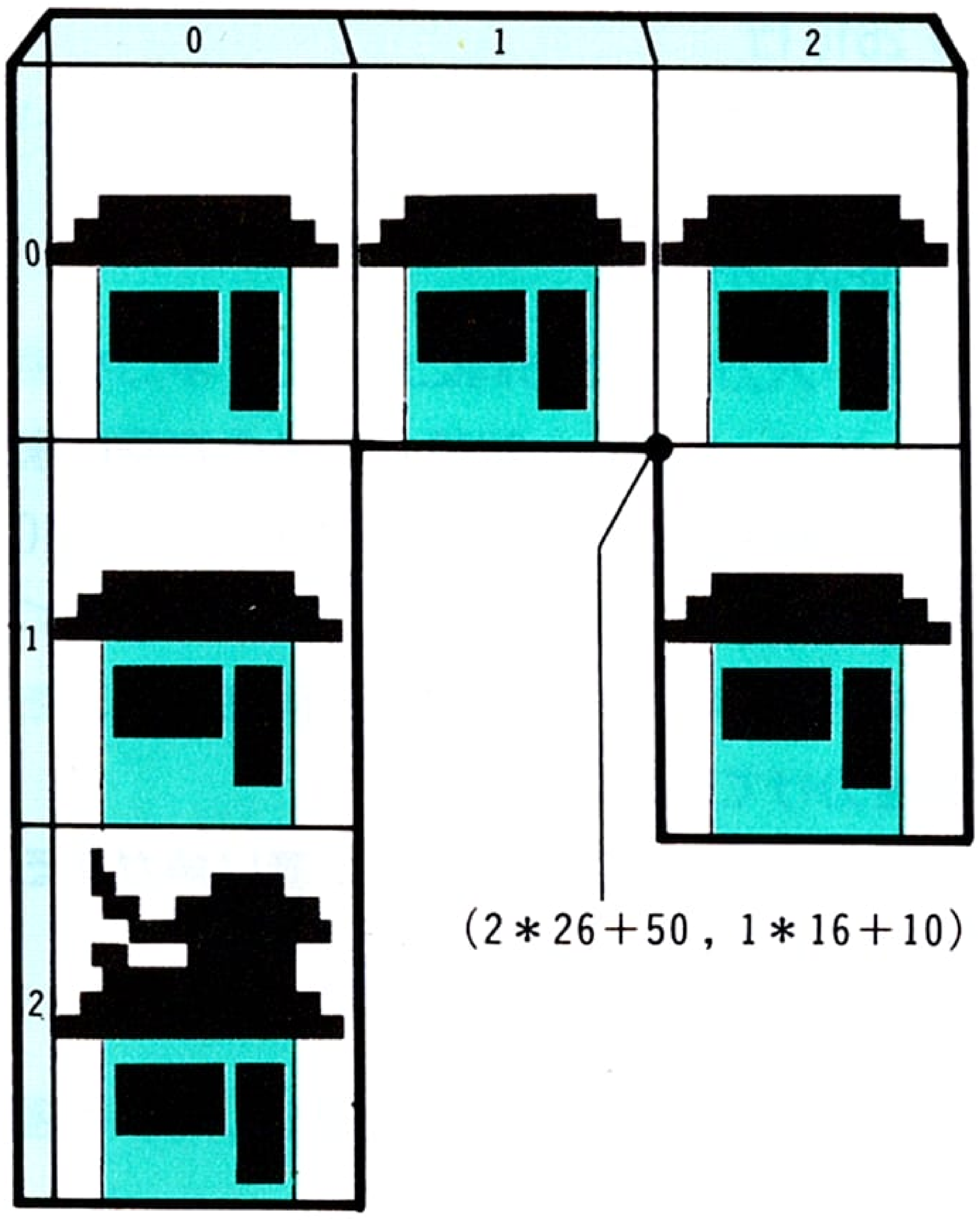


Figure 6

しています（*Figure 6* 参照）.

● 2040～2110 行

2060 行で家のあるところを判断して，乱数でそのどこかに明かりをつけ，このデータを変数 LX (I)，LY (I) に取り込んでおきます．この明かりが，ネコを追いかけるわけです．

● 2150～2210 行

ネコの開始位置を示し，4 回点滅させせます．

● 2230，2240 行

[STOP] 以外の任意のキーを押すと，ゲームがスタートするようにしています．

● 2260 行

スコア PT を 0 にしています．

● 2280，2290 行

点数が入るたびに点数を書き換えて表示します．

● 2310～2590 行

ここで，変数 LMAX 個の明かり（この場合は LMAX=2）が，ネコを追いかけています．この追いかけ方に注目してください．

2370～2420 行の FOR～NEXT ループでは，上を 1，下を 2，左を 3，右を 4 として，それぞれ明かりの位置とネコの x 軸方向，y 軸方向の距離を計算し，x 軸方向，y 軸方向のおのおのの向きにおいて最小となる方向 R を，変数 RX，RY に代入します．2380，2390 行が x 軸方向，2400，2410 行が y 軸方向について距離の計算をし，判定する部分です（*Figure 7* 参照）.

2430 行でこの RX の方向に移動したときの新しい場所を XN，YN とします．そし

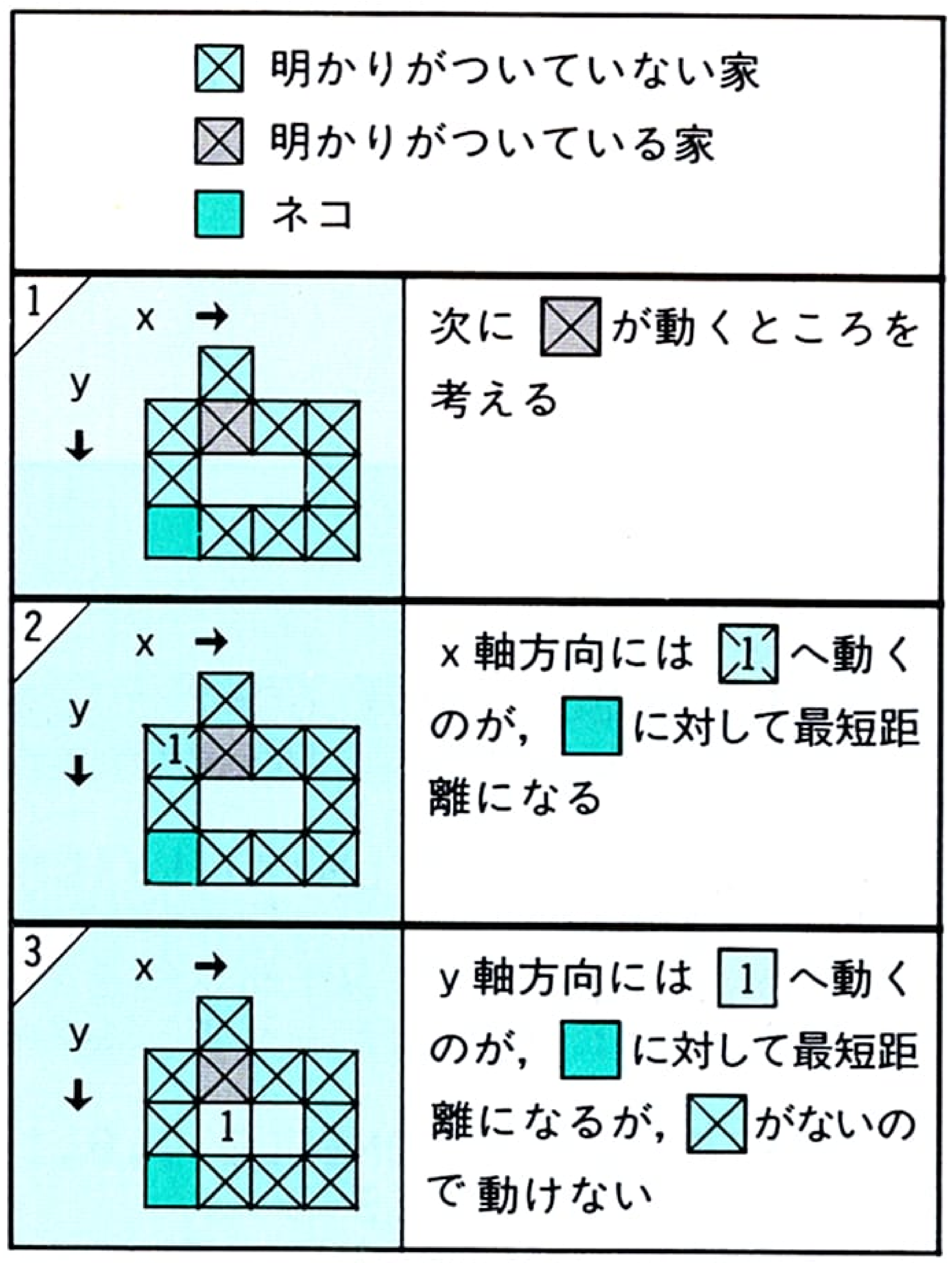


Figure 7

て，その位置に家があれば移動できるので，それを 2440 行で判断します．ただし，ここで RND>0.5 になる場合は，あえてその方向（つまり x 軸方向の距離が最小になる方向）には動かさず，次の y 軸方向が最小になる方向に実行を移します．2460 行は，同様のことを y 軸方向に対して行っています．

2480〜2510 行は，明かりが次に動く位置を乱数で決定している部分です．もし，仮にネコとの距離が最小になる方向に明かりを移動したくても，そこに家がない場合，乱数で他の方向に明かりを移動させます．また，2 つある明かり（1060 行の LMAX＝2 で設定している）のうち最初の 1 つは，常にこの乱数のみで方向を決定しています．2360 行でその判断をしています．

明かりが動く方向が決定したら，2530 〜 2570 行で明かりを移動させます．このとき，2580 行で，明かりとネコが重なったかどうか，判断しています．

- 2610 行

ネコを動かすためのキー入力を行っています．

- 2630〜2750 行

キー入力された方向に対して，ネコを動かします．しかし，2660 行でその方向に家がない場合は，キー入力を無効にして，2310 行に戻ります．また，2710，2720 行で動く方向 R によって，右向き，左向きのネコを描き分けています．

2740 行は，ネコが動いた位置に魚があるかないかを判断している部分です．

- 2790〜2870 行

ネコが捕まった場合の処理をしている部分です．

- 2910〜3010 行

ネコが魚をすべて取った場合の処理です．

P.S.　このプログラムの制作に協力してくださった，武田君，金子さんどうもありがとう．

List 6.4

```
1000 ' List 6.4
1010 '
1020 DEFINT A-Z
1030 BC1=0  : BC2=4  : CC=7   : FC=1
1040 LOOF=2 : WALL=7 : DOOR=1 : LC=3
1050 PS=100
1060 LMAX=2 : DIM LX(LMAX),LY(LMAX)
1070 KY$="5213"
1080 XS(1)=0  : XS(2)=0 : XS(3)=-1 : XS(4)=1
1090 YS(1)=-1 : YS(2)=1 : YS(3)=0  : YS(4)=0
1100 '
1110 WIDTH 80,25 : CONSOLE ,,0,1 : COLOR 7
1120 SCREEN 0,0  : CLS 3
1130 RANDOMIZE VAL(RIGHT$(TIME$,2))
1140 DIM HOUSE(100),CATR(40),CATL(40),FISH(40),D$(20,10)
```

（1030〜1090 行）変数の定義と初期設定

```
1150 '
1160 '
1170 '                                配列に家,ネコ,魚を取り込むために背景を描く
1180 LINE (0,0)-(103,15),BC1,BF
1190 FOR X=0 TO 103 STEP 2 : LINE (X,0)-(X,15),BC2 : NEXT X
1200 LOCATE 0,5 : PRINT "Wait a minute..."
1210 '
1220 '
1230 '                                家を描き,そのデータを配列に取り込む
1240 LINE (4,6)-(0,8),LOOF : LINE -(23,8),LOOF
1250 LINE -(19,6),LOOF : LINE -(4,6),LOOF : PAINT (10,7),LOOF
1260 LINE (4,9)-(19,15),WALL,BF
1270 LINE (15,10)-(18,14),DOOR,BF
1280 LINE (5,10)-(13,12),0,BF
1290 GET (0,0)-(25,15),HOUSE
1300 '
1310 '
1320 '                                右向きのネコを描き,そのデータを配列に取り込む
1330 RESTORE 1450
1340 READ X,Y : POINT (25+X,Y)
1350 FOR I=1 TO 17 : READ X,Y : LINE -(25+X,Y),CC : NEXT I
1360 GET (26,0)-(51,5),CATR
1370 '
1380 '
1390 '                                左向きのネコを描き,そのデータを配列に取り込む
1400 RESTORE 1450
1410 READ X,Y : POINT (75-X,Y)
1420 FOR I=1 TO 17 : READ X,Y : LINE -(75-X,Y),CC : NEXT I
1430 GET (52,0)-(77,5),CATL
1440 '
1450 DATA  3,0,  7,2, 15,2, 17,2, 17,1, 19,2, 21,3, 19,3, 14,3
1460 DATA 17,5, 15,5, 14,3,  8,3,  7,5,  5,5,  7,3,  2,0,  3,0
1470 '
1480 '
1490 '                                魚を描き,そのデータを配列に取り込む
1500 RESTORE 1560
1510 READ X,Y : POINT (75+X,Y)
1520 FOR I=1 TO 8 : READ X,Y : LINE -(75+X,Y),FC : NEXT I
1530 PAINT (75+15,3),FC
1540 GET (78,0)-(103,5),FISH
1550 '
```

```
1560 DATA 3,1, 7,3, 16,1, 22,3, 16,4, 7,3, 3,4, 7,3, 3,1
1570 '
1580 '
1590 '
1600 CLS : LINE (0,0)-(639,199),BC1,BF
1610 FOR X=0 TO 639 STEP 2 : LINE (X,0)-(X,199),BC2 : NEXT X
1620 '
1630 '
1640 '
1650 RESTORE 1770
1660 FMAX=0: PC=0
1670 FOR Y=0 TO 9
1680   READ D$
1690   FOR X=0 TO 19
1700     L$=MID$(D$,X+1,1)
1710     IF L$="*" THEN D$(X,Y)="*"
1720     IF L$="@" THEN D$(X,Y)="*" : CX=X: CY=Y
1730     IF L$="1" THEN D$(X,Y)="1" : FMAX=FMAX+1 : PC=PC+1
1740   NEXT X
1750 NEXT Y
1760 '
1770 DATA "1****  1****1*******"
1780 DATA "*   *   * *   *    *"
1790 DATA "********* **1****1 *"
1800 DATA "*   *     *   *    *"
1810 DATA "*** 1 1***@*********"
1820 DATA "* *   *  *   *     *"
1830 DATA "* *****  **1***    *"
1840 DATA "*** * ****   *******"
1850 DATA "* * 1   *    *1*   *"
1860 DATA "**1   1**********1**"
1870 '
1880 ' This map is arranged by Tsunehiro Take-chan
1890 '
1900 '
1910 '
1920 '
1930 FOR Y=0 TO 9
1940   FOR X=0 TO 19
1950     IF D$(X,Y)<>"*" AND D$(X,Y)<>"1" GOTO 1990
1960     GX=X*26+50: GY=Y*16+10
1970     PUT (GX,GY),HOUSE,PSET
1980     IF D$(X,Y)="1" THEN PUT (GX,GY),FISH,PSET
1990   NEXT X
2000 NEXT Y
```

背景を中間色で描く（1600〜1610）

地図のデータを配列に取り込む（1650〜1750）

地図のデータ（1770〜1860）

家，魚を画面に表示する（1930〜2000）

```
2010 '
2020 '
2030 '
2040 FOR I=1 TO LMAX
2050   X=INT(RND*20) : Y=INT(RND*10)
2060   IF D$(X,Y)<>"*" AND D$(X,Y)<>"1" GOTO 2050
2070   IF X=CX AND Y=CY GOTO 2050
2080   LX(I)=X    : LY(I)=Y
2090   GX=X*26+55 : GY=Y*16+20
2100   LINE (GX,GY)-(GX+8,GY+2),LC,BF
2110 NEXT I
2120 '
2130 '
2140 '
2150 FOR I=1 TO 4
2160   GX=CX*26+50 : GY=CY*16+10
2170   PUT (GX,GY),HOUSE,PSET
2180   FOR T=1 TO 100 : NEXT T : BEEP 1 : BEEP 0
2190   PUT (GX,GY),CATL,PSET
2200   FOR T=1 TO 100 : NEXT T
2210 NEXT I
2220 '
2230 CLS : LOCATE 25,24 : PRINT "Hit any key to start !!";
2240 DUMMY$=INPUT$(1) : CLS
2250 '
2260 PT=0
2270 '
2280 *SCORE
2290 LOCATE 30,24 : PRINT USING "Point=####";PT;
2300 '
2310 *MAIN
2320 GOSUB *BELL1
2330 IF INKEY$<>"" GOTO 2330
2340 FOR I=1 TO LMAX
2350   RX=1 : RY=1 : DISX=20 : DISY=10
2360   IF I<2 GOTO 2480
2370   FOR R=1 TO 4
2380     DIS=ABS(CX-(LX(I)+XS(R)))
2390     IF DIS<=DISX THEN RX=R : DISX=DIS
2400     DIS=ABS(CY-(LY(I)+YS(R)))
2410     IF DIS<=DISY THEN RY=R : DISY=DIS
2420   NEXT R
2430   XN=LX(I)+XS(RX) : YN=LY(I)+YS(RX)
2440   IF (D$(XN,YN)="*"OR D$(XN,YN)="1")AND RND>.5 GOTO 2530
2450   XN=LX(I)+XS(RY) : YN=LY(I)+YS(RY)
2460   IF D$(XN,YN)="*" OR D$(XN,YN)="1" GOTO 2530
```

窓の明かりを表示する（2040〜2110）

ネコを描く（2150〜2210）

ゲーム開始時のデモンストレーション（2230〜2240）

スコア表示（2280〜2290）

メインルーチン（2310〜）

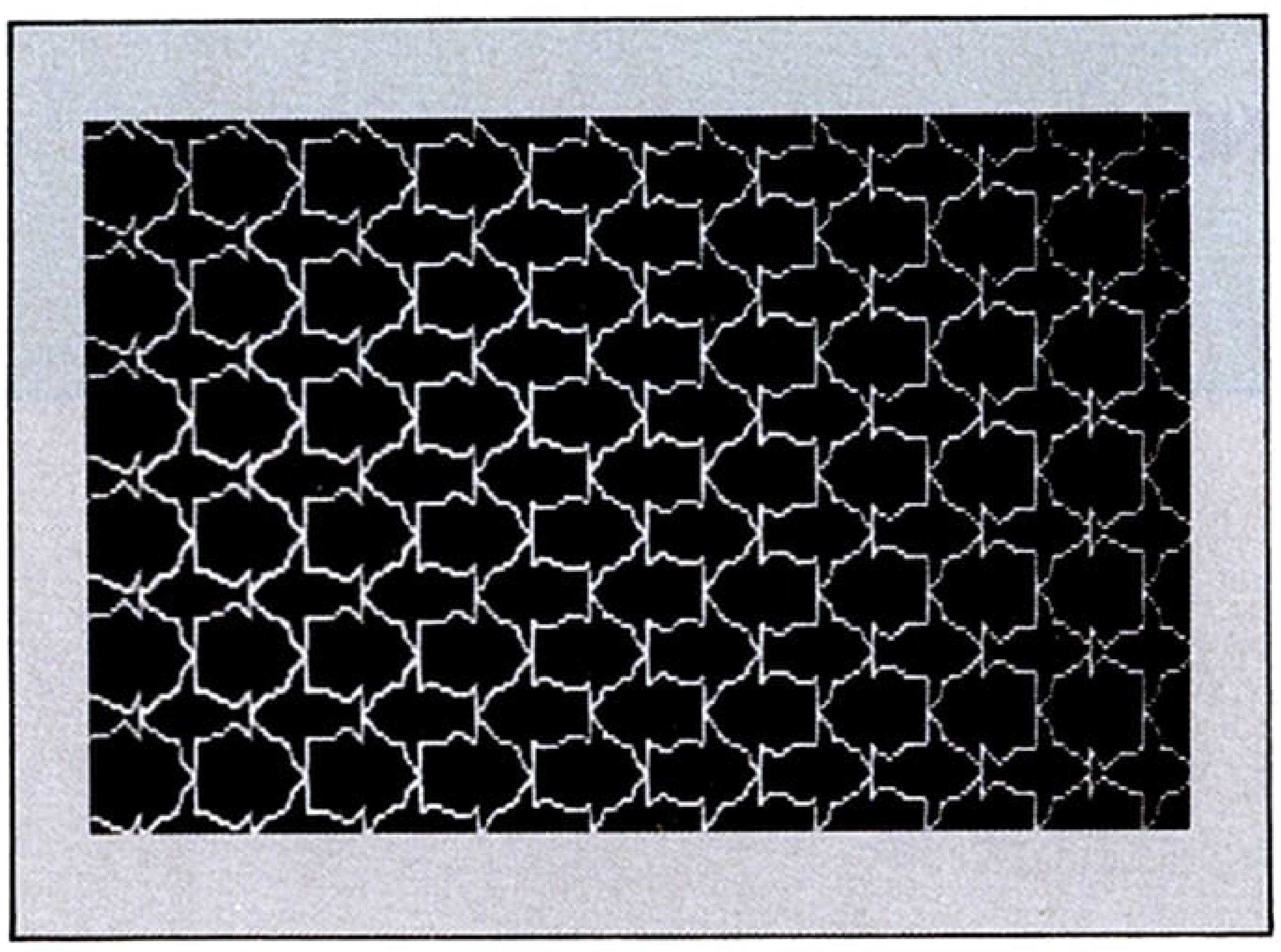
```
2470   '
2480   R=INT(RND*4)+1
2490   XN=LX(I)+XS(R) : YN=LY(I)+YS(R)
2500   IF XN<0 OR YN>19 OR YN<0 OR YN>9 GOTO 2480
2510   IF D$(XN,YN)<>"*" AND D$(XN,YN)<>"1" GOTO 2480
2520   '
2530   GX=LX(I)*26+55 : GY=LY(I)*16+20
2540   LINE (GX,GY)-(GX+8,GY+2),0,BF
2550   LX(I)=XN : LY(I)=YN
2560   GX=XN*26+55 : GY=YN*16+20
2570   LINE (GX,GY)-(GX+8,GY+2),LC,BF
2580   IF LX(I)=CX AND LY(I)=CY GOTO *CAUGHT
2590 NEXT I
2600 '
2610 IN$=INKEY$: IF IN$="" THEN IN$=IO$
2620 '
2630 R=INSTR(KY$,IN$) : IF R<1 OR R>4 GOTO *MAIN
2640 XN=CX+XS(R) : YN=CY+YS(R)
2650 IF XN<0 OR XN>19 OR YN<0 OR YN>9 GOTO *MAIN
2660 IF D$(XN,YN)<>"*" AND D$(XN,YN)<>"1" GOTO *MAIN
2670 GX=CX*26+50 : GY=CY*16+10
2680 PUT (GX,GY),HOUSE,PSET
2690 CX=XN : CY=YN : IO$=IN$
2700 GX=CX*26+50 : GY=CY*16+10
2710 IF R=1 OR R=4 THEN PUT (GX,GY),CATR,PSET
2720 IF R=2 OR R=3 THEN PUT (GX,GY),CATL,PSET
2730 '
2740 IF D$(XN,YN)="1" THEN PC=PC-1 : PT=PT+PS : GOSUB *BELL2 :
                  D$(XN,YN)="*" : IF PC=0 GOTO *FINISH
2750 GOTO *SCORE
2760 '
2770 '
```

```
2780 ' ネコが捕まったときの処理
2790 *CAUGHT
2800 GOSUB *BELL3 : CLS
2810 LOCATE 22,24 : PRINT "The Cat has caught by the people !!";
2820 FOR T=1 TO 2000 : NEXT T : CLS
2830 LOCATE 22,24 : PRINT "Continue this game ? (y/n)";
2840 I$=INPUT$(1)
2850   IF I$="y" OR I$="Y" THEN CLS : GOTO 1930
2860   IF I$="n" OR I$="N" THEN END
2870 GOTO 2840
2880 '
2890 '
2900 ' ネコが魚を全部とったときの処理
2910 *FINISH
2920 GOSUB *BELL4 : CLS
2930 LOCATE 22,24 : PRINT "The Cat had eaten all fish !!";
2940 FOR T=1 TO 2000 : NEXT T : CLS
2950 LOCATE 22,24 : PRINT USING "ｺﾞｸﾛｳｻﾏ ﾃﾞｽ｡  Point=####";PT;
2960 FOR T=1 TO 2000 : NEXT T : CLS
2970 LOCATE 22,24 : PRINT "Continue this game ? (y/n)";
2980 I$=INPUT$(1)
2990   IF I$="y" OR I$="Y" THEN CLS : GOTO 1650
3000   IF I$="n" OR I$="N" THEN END
3010 GOTO 2980
3020 '
3030 '
3040 '
3050 *BELL1 : BEEP 1 : BEEP 0 : RETURN
3060 *BELL2 : BEEP 1 : FOR TT=1 TO 10 : NEXT TT : BEEP 0 : RETURN
3070 *BELL3 : BEEP : RETURN
3080 *BELL4
3090 FOR II=1 TO 3
3100   BEEP 1 : FOR TT=1 TO 30 : NEXT TT : BEEP 0
3110   FOR TT=1 TO 100 : NEXT TT
3120 NEXT II : RETURN
```

メタモルフォーゼかな?

コンピュータ・グラフィックスという，この新しいテクニックは，今まではあまりなかった，あるいは不可能であったことができるということから，アートとしての無限の可能性を秘めているといえます．むかしむかしの画家たちが，パソコンを駆使してコンピュータ・グラフィックスで絵を描いたらいったいどんな作品ができるのでしょうね．例えば，エッシャーというオランダの版画家を御存知ですか？ 観る人にとても不思議な印象を与える彼の一連の作品には，心理学や数学の要素がたくさん入っています．２次元図形であるはずの六角形が，いつのまにか，生きたトカゲになってしまう絵とか，果てしなく登り続ける階段の絵など，エッシャーの作品はまさに視覚を刺激するものばかりです．

「メタモルフォーゼ」は，特に有名な彼の作品群です．ここでは，その絵をちょっとまねてアートしてみました．この「メタモルフォーゼ」は，その言葉どおり，鳥が魚へ，昼が夜へ，黒鳥が白鳥へというようにあるものがある別のものへと変化する様子を一枚の絵の中で表現しています．このプログラムは，その変形(メタモルフォーゼ)の第一段階として左から右へと変化していく魚と右から左へと変化していく魚が交互に組み合わさっている絵を描いてみたものです．この変化の仕組みは意外に単純です．まず出発点となる図形を用意します．またそれとは別に，最終的な図形のデータも用意します．そして最初の図形からほんのすこしだけ最後の図形に近づいた形，それよりもさらにすこし最後の図形に近づいた形………を順に計算し，次々とそれを横に並べていきます．簡単なようですが，図形のデータを工夫すればこのプログラムのようにおもしろいグラフィックができあがるのです．エッシャーの絵に限らずいろいろな絵画をまねて，あなたもコンピュータ・グラフィックスでアートしてみてはいかがでしょう．

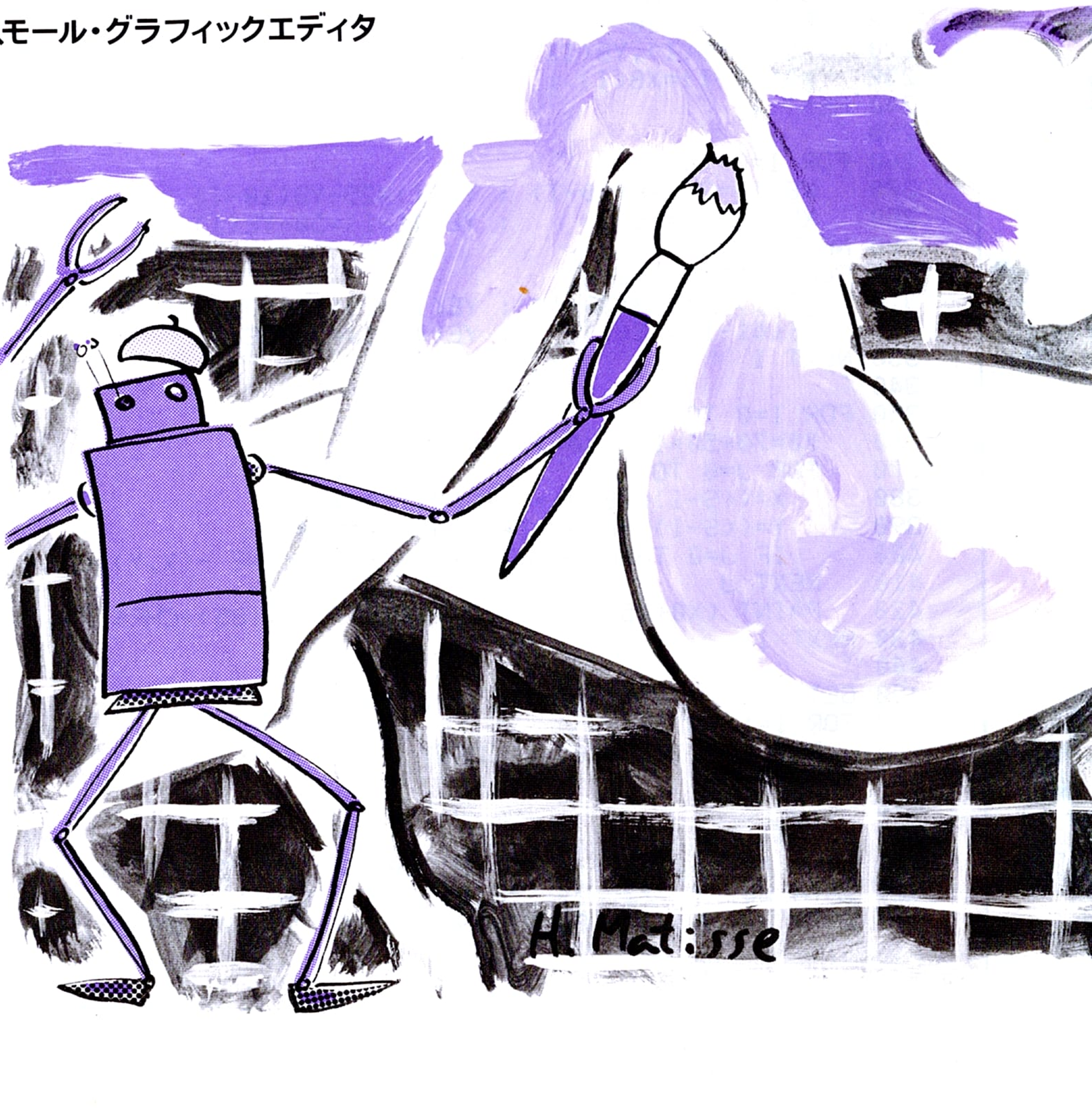


ついでですが，このグラフィックのプログラムを作ってみてエッシャーのメタモルフォーゼの作品群は実にコンピュータで描きやすいということに気がつきました．もしかしたら，エッシャーは手計算であんな版画を作っていたのかもしれません．彼は，彼の時代におけるコンピュータ・グラフィックデザイナーだったのです．

```
100 ' List Ex.5
110 '
120 CONSOLE ,,0,1 : SCREEN 0,0 : CLS 3
130 DIM XB(41),YB(41),XE(41),YE(41),GD(1230)
140 S=9 : C=7
150 '
160 LINE (69,24)-(71+50*10,26+25*6),7,B
170 LINE (70,25)-(71+50*10-1,26+25*6-1),1,BF
180 PAINT (0,0),CHR$(255)+CHR$(85)+CHR$(85),7
190 '
200 RESTORE *GDATA
210 FOR I=0 TO 20
220   READ XB(I),YB(I)
230   XB(41-I)=XB(I) : YB(41-I)=50-YB(I)
240 NEXT I
250 '
260 FOR I=0 TO 10
270   XE(I)=25-XB(10-I) : YE(I)=25-YB(10-I)
280   XE(41-I)=XE(I)    : YE(41-I)=50-YE(I)
290 NEXT I
300 FOR I=11 TO 20
310   XE(I)=75-XB(30-I) : YE(I)=25-YB(30-I)
320   XE(41-I)=XE(I)    : YE(41-I)=50-YE(I)
330 NEXT I
340 '
350 FOR I=0 TO S
360   X0=70+50*I : Y0=25
370   FOR J=0 TO 41
380     X=((S-I)*XB(J)+I*XE(J))/S+X0
390     Y=((S-I)*YB(J)+I*YE(J))/S/2+Y0
400     IF J=0 THEN POINT (X,Y) ELSE LINE -(X,Y),C
410   NEXT J
420   PAINT (X0+25,Y0+12),CHR$(85)+CHR$(85)+CHR$(0),7
430 NEXT I
440 '
450 GET (70,Y0)-(70+50*10,Y0+25),GD
460 FOR I=1 TO 5
470   PUT (70,I*25+Y0),GD,PSET
480 NEXT I
490 '
500 *GDATA
510 DATA  0,25,  0, 9,  5, 8, 10, 7, 13, 5, 15, 4, 18, 2
520 DATA 20, 3, 22, 5, 25,10, 25, 0, 26, 2, 28, 3, 30, 5
530 DATA 32, 6, 34, 8, 37, 9, 42,10, 43,17, 45,20, 50,25
```

Appendix

スモール・グラフィックエディタ





スモール・グラフィックエディタ

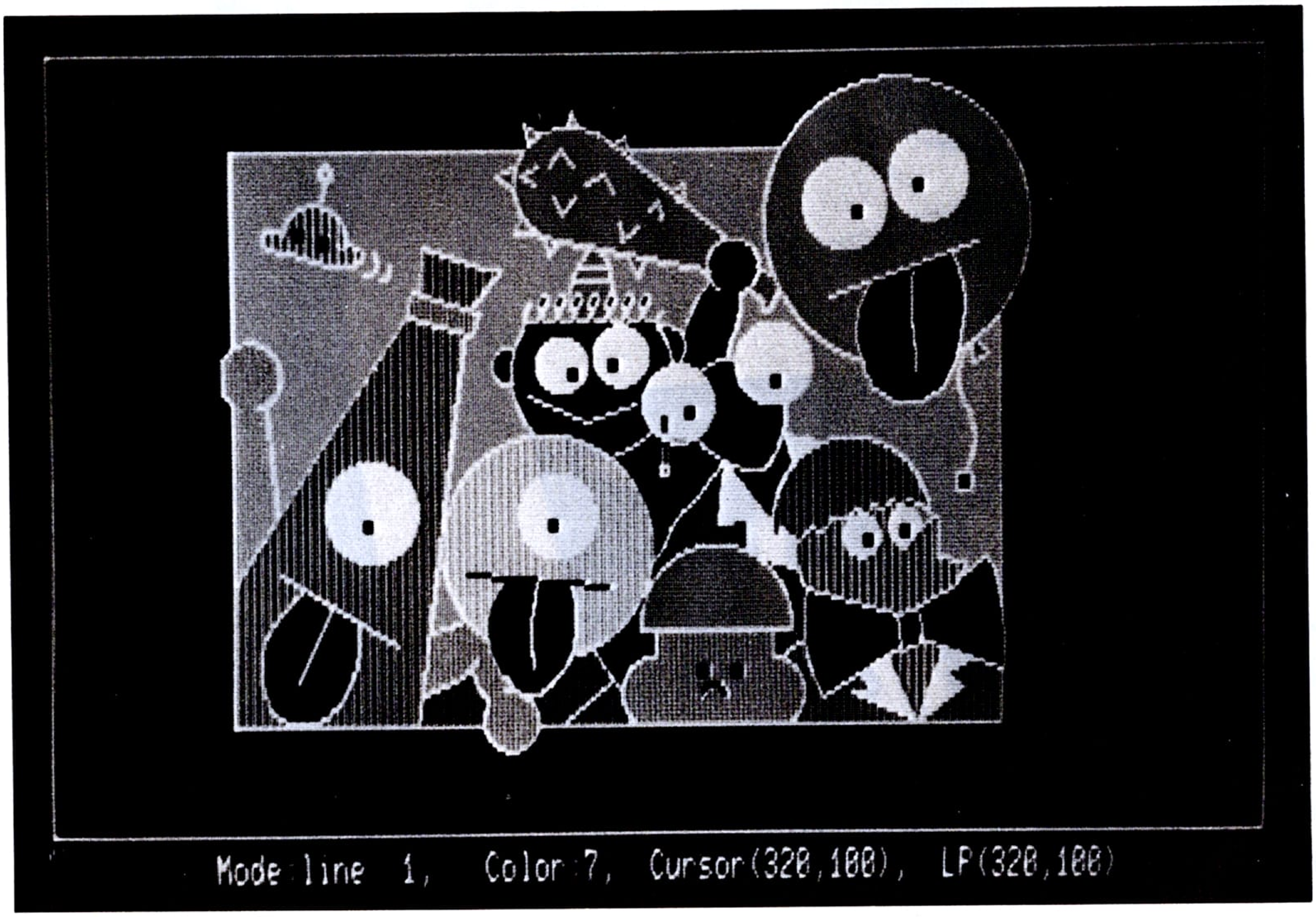
スモール・グラフィックエディタは，コンピュータが絵具と筆とキャンパスの役目を果たしてくれるグラフィックツールです．あなたの感性にまかせて筆を進めていけば，水彩や油彩などとは全く異なった新しい感覚の絵を描くことができます．さっそくこのスモール・グラフィックエディタを使ってみてください．

利用法

このツールは，3 種類の太さを持った筆と，3 種類の太さに描けるブラシを使って絵を描いていくものです．

◇カーソルについて

カーソルは 2 つあります．1 つは白色で縦



の線が太いもの，もう１つは水色の十字のものです．以後，前者をメインカーソル，後者をLPカーソルと呼びます．

　すべての描画は，メインカーソルを中心に行います．画面の下端に，モードの設定などの表示があります．このうち“Mode：”の後は，初期状態では“line”ですが，Bキーを押すと“brush”に，Lキーを押すと“line”にと変わります．ブラシモードではスプレーを吹きつけたような効果が得られます．筆やブラシの太さは，“line”あるいは“brush”の次に表示されている数字で，1〜3まで選ぶことができます．LPカーソルは，直線や四角形，円を描くときに使います．

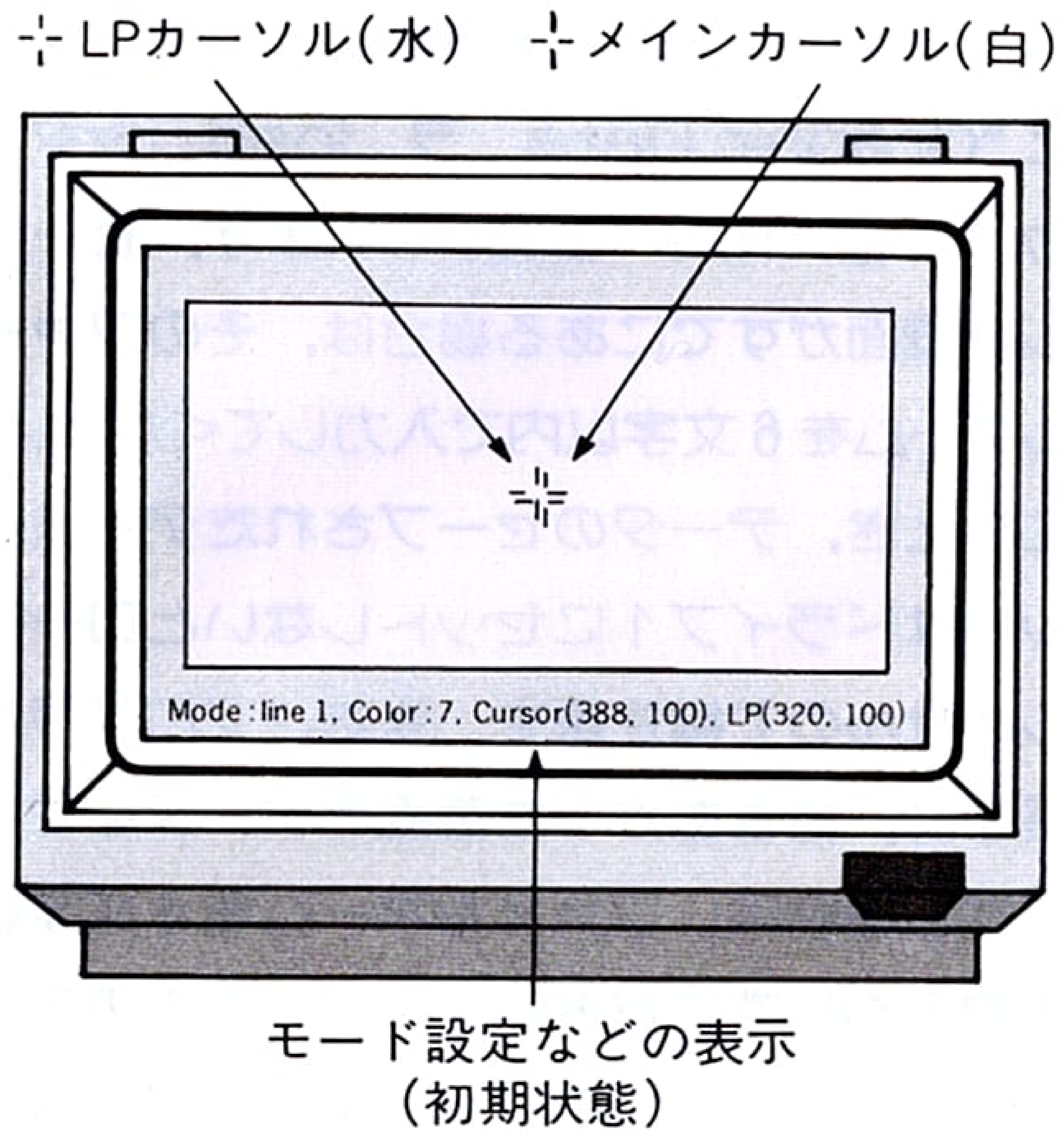


Figure 1

◇キー操作

　ここで，すべてのキー操作を説明します．

B：ブラシモードの選択

L：ラインモードの選択

↑↓：色の選択

←→：筆やブラシの太さの選択．←で細く，→で太くなる

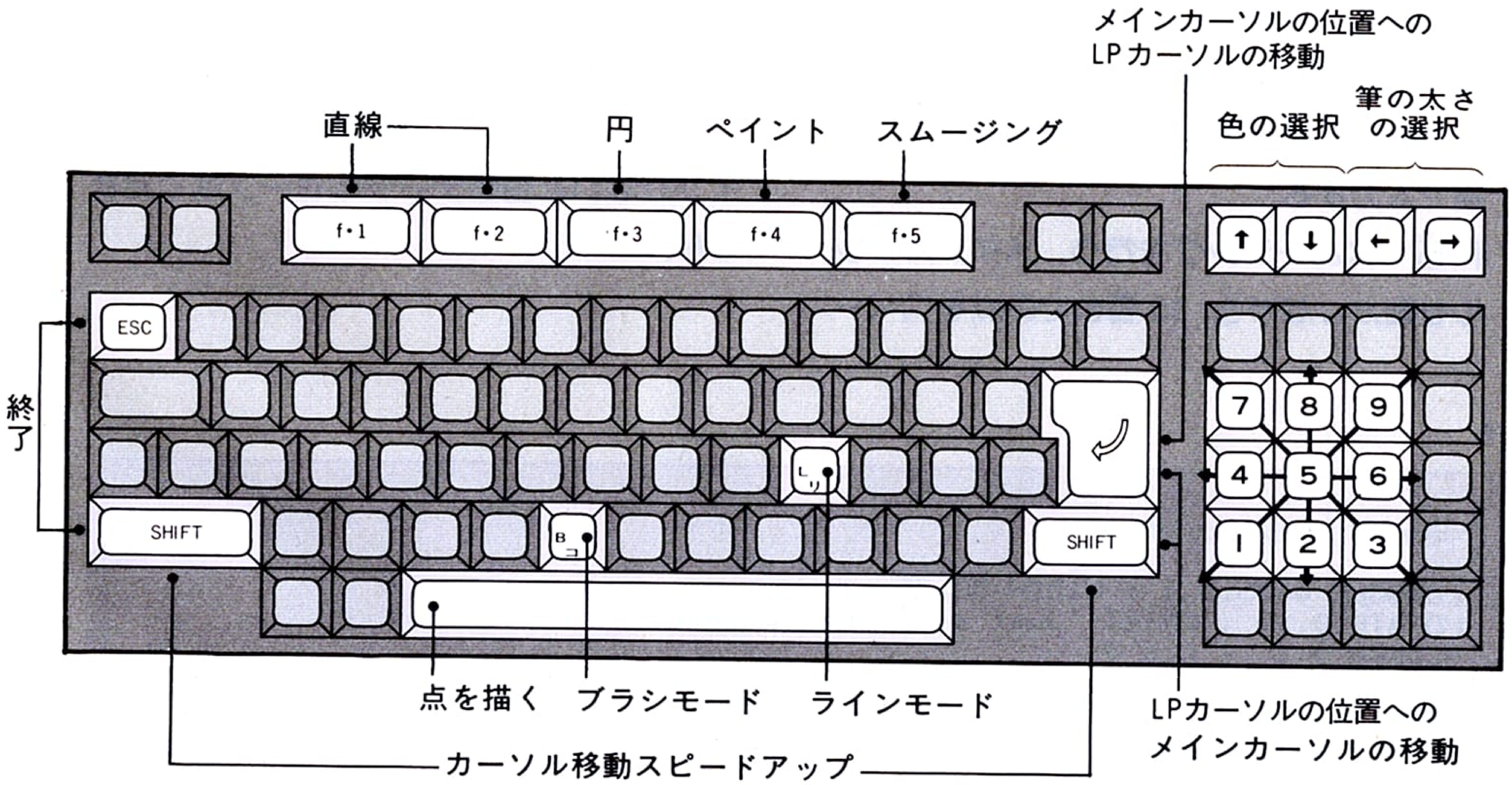


Figure 2

テンキー：メインカーソルの移動．5 を中心として 8 方向に移動する

SHIFT ＋テンキー：メインカーソルを17ドットずつ移動させる

スペースキー：メインカーソルの位置に，そのカーソルの情報に従って，点あるいはブラシ状の点を描く

RETURN ：メインカーソルの位置にLPカーソルを移動させる

SHIFT ＋ RETURN ：LPカーソルの位置にメインカーソルを移動させる

F１：メインカーソルとLPカーソル間を直線で結ぶ．その後メインカーソル位置にLPカーソルを移動させる

F２：F１ と同様に 2 点を結ぶが，LPカーソルを移動させない

F３：LPカーソルを中心とし，メインカーソルを通る円を描く

F４：メインカーソルを中心としてペイントを行う．ノーマルなペイントと，タイルペイントが選択できる．タイルペイントでは三原色の混合の比率を，それぞれ 0～4 の値で入力する

F５：スムージングを行う．メインカーソルを動かしながら，10 個までの点をスペースキーで指定していく．好みの個数で，RETURN キーを押すと，指定した点を通るなめらかな曲線が描かれる．ただし，同じ位置で 2 点を指定してはいけない

SHIFT ＋ESC：終了する

このプログラムを実行すると，まず始めに“Graphic Data ヲ load シマスカ ？（y／n)？”と表示されます．セーブした画面がすでにある場合は，そのファイルネームを 6 文字以内で入力してください．このとき，データのセーブされたディスケットはドライブ 1 にセットしないとロードしませんので御注意を．また，プログラム終了時，描きあがった画面のデータをセーブする場合は，ファイルネームを 6 文字以内で入力してください．ドライブ 1 のディスケット上に 4 つのファイルとして記録されます．なお，データのセーブには 2 分以上かかります．

以上でキー操作の説明を終わります．

なお，このプログラムはディスク・ドライブがないと画面データのセーブはできません．

List A.1

```
1000 ' Small Graphic Editor
1010 '
1020  CONSOLE 0,25,0,1 : WIDTH 80,25
1030  LOCATE 0,10 : PRINT "Just a moment, please."
1040  SCREEN 0,0  : CLS 2
1050 '
1060  PRINT "Graphic Data ヲ load シマスカ ? (y/n)";
1070  INPUT I$ : IF I$="Y" OR I$="y" GOTO *GDLOAD
1080 '
```

```
1090 *INIT
1100  RANDOMIZE VAL(RIGHT$(TIME$,2))
1110  DIM CR(26),LP(26)
1120  DN=3 : DIM GX(DN),GY(DN),B(3,11*DN),BR(3,11*DN)
1130  DIM L2(4),L2R(4),L3(8),L3R(8),D1(4),D2(4),D3(4)
1140  DIM X(73),Y(73),D(73),MX(73,1),MY(73,1)
1150  IC=7 : IY=24
1160  CX=320 : CY=100 : LPX=CX : LPY=CY
1170  MODE$="line" : THICK=1 : CL=7
1180  DEF FNNUM(X,Y)=(((X+7)¥8)*Y*3+4)¥4+1
1190 '
1200  GOSUB *GETDATA
1210  GOSUB *TILE
1220  LINE (24,12)-(616,188),7,B
1230  GOSUB *INFORM
1240  GOSUB *CURSOR : GOSUB *LP
1250 '
1260 *MAIN
1270  GOSUB *KEYINPUT
1280  IF P8=251 OR PA=251 THEN GOSUB *THICK    :  GOTO *MAIN
1290  IF P8=253 OR PA=253 THEN GOSUB *CCHANGE :  GOTO *MAIN
1300 '
1310  GOSUB *TENKEY
1320  IF P8=191 GOTO *SHIFT
1330  IF P1=127 THEN GOSUB *LPTOCR  : GOTO *MAIN
1340  IF P2=251 THEN GOSUB *BMODE   : GOTO *MAIN
1350  IF P3=239 THEN GOSUB *LMODE   : GOTO *MAIN
1360  IF P9=191 THEN GOSUB *DPUT    : GOTO *MAIN
1370  IF P9=247 THEN GOSUB *CIRCLEN : GOTO *MAIN
1380  IF P9=239 THEN GOSUB *PAINTER : GOTO *MAIN
1390  IF P9=223 THEN GOSUB *SMOOTH  : GOTO *MAIN
1400  IF P9=253 THEN F=-1: GOSUB *LINE1: GOTO *MAIN
1410  IF P9=251 THEN GOSUB *LINE1   : GOTO *MAIN
1420  GOTO *MAIN
1430 '
1440 *SHIFT
1450  IF P1=127 THEN GOSUB *CRTOLP
1460  IF P9=127 GOTO *GDSAVE
1470  GOTO *MAIN
1480 '
1490 *TENKEY
1500  IF P0=239 THEN SX=-1 : SY=0  : GOSUB *MOVE
1510  IF P0=127 THEN SX=-1 : SY=-1 : GOSUB *MOVE
1520  IF P1=254 THEN SX=0  : SY=-1 : GOSUB *MOVE
1530  IF P1=253 THEN SX=1  : SY=-1 : GOSUB *MOVE
1540  IF P0=191 THEN SX=1  : SY=0  : GOSUB *MOVE
1550  IF P0=247 THEN SX=1  : SY=1  : GOSUB *MOVE
1560  IF P0=251 THEN SX=0  : SY=1  : GOSUB *MOVE
1570  IF P0=253 THEN SX=-1 : SY=1  : GOSUB *MOVE
1580  RETURN
1590 '
1600 *KEYINPUT
1610  P0=INP(0) : P1=INP(1) : P2=INP(2)  : P3=INP(3)
1620  P8=INP(8) : P9=INP(9) : PA=INP(10) : PB=INP(11)
1630  IF INKEY$<>"" GOTO 1630
1640  RETURN
1650 '
```

押されているキーに対応する処理に飛ぶ (1260–1420)

シフトキーが押されているときの処理 (1440–1470)

テンキーと対応する方向へカーソルを動かす (1500–1570)

```
1660 *CURSOR
1670  PUT (CX-9,CY-5),CR,XOR : RETURN
1680 '
1690 *LP
1700  PUT (LPX-9,LPY-5),LP,XOR : RETURN
1710 '
1720 *INFORM
1730  FORM1$="Mode:&   & #_,   Color:#_,
1740  FORM2$="Cursor(###_,###)_,  LP(###_,###)"
1750  CLS : LOCATE 13,IY : COLOR IC
1760  PRINT USING FORM1$+FORM2$;MODE$,THICK,CL,CX,CY,LPX,LPY;
1770  COLOR 7 : RETURN
```

画面上の情報を文字で表示する

```
1780 '
1790 *THICK
1800  IF PA=251 AND THICK>1 THEN THICK=THICK-1
1810  IF P8=251 AND THICK<3 THEN THICK=THICK+1
1820  GOSUB *INFORM
1830  RETURN
```

筆の太さを変える

```
1840 '
1850 *MOVE
1860  IF P8=191 THEN SX=SX*17 : SY=SY*17
1870  IF CX+SX<11 OR CX+SX>629 THEN RETURN
1880  IF CY+SY<6 OR CY+SY>194  THEN RETURN
1890  GOSUB *CURSOR
1900  CX=CX+SX : CY=CY+SY : GOSUB *CURSOR
1910  GOSUB *INFORM
1920  RETURN
```

カーソルを移動させる

```
1930 '
1940 *LPTOCR
1950  GOSUB *LP
1960  LPX=CX : LPY=CY : GOSUB *LP
1970  GOSUB *INFORM
1980  RETURN
```

LPカーソルをメインカーソルの位置へ動かす

```
1990 '
2000 *CRTOLP
2010  GOSUB *CURSOR
2020  CX=LPX : CY=LPY : GOSUB *CURSOR
2030  GOSUB *INFORM
2040  RETURN
2050 '
2060 *CCHANGE
2070  IF PA=253 AND CL>0 THEN CL=CL-1
2080  IF P8=253 AND CL<7 THEN CL=CL+1
2090  GOSUB *INFORM
2100  RETURN
2110 '
2120 *BMODE
2130  MODE$="brush"
2140  GOSUB *INFORM
2150  RETURN
2160 '
2170 *LMODE
2180  MODE$="line"
2190  GOSUB *INFORM
2200  RETURN
2210 '
```

```
2220 *DPUT
2230  DX=CX : DY=CY
2240  GOSUB *CURSOR : GOSUB *LP
2250  GOSUB *DOT
2260  GOSUB *CURSOR : GOSUB *LP
2270  RETURN
2280 '
2290 *DOT
2300  IF MODE$='brush' THEN GOSUB *BRUSH
2310  IF MODE$='line'  THEN GOSUB *DLINE
2320  RETURN
2330 '
2340 *DLINE
2350  IF DX<9 OR DX>629 OR DY<4 OR DY>194 THEN RETURN
2360  IF THICK=1 THEN PSET (DX,DY),CL
2370  IF THICK=2 THEN PUT (DX-4,DY-2),L2R,AND,7,0 : PUT (DX-4,DY-
2),L2,OR,CL,0
2380  IF THICK=3 THEN PUT (DX-9,DY-4),L3R,AND,7,0 : PUT (DX-9,DY-
4),L3,OR,CL,0
2390  RETURN
2400 '
2410 *BRUSH
2420  IF DX<11 OR DX>627 OR DY<5 OR DY>193 THEN RETURN
2430  RN=INT(RND*DN)
2440  PUT (DX-GX(THICK),DY-GY(THICK)),BR(THICK,RN*GX(THICK)),AND,
7,0
2450  PUT (DX-GX(THICK),DY-GY(THICK)),B(THICK,RN*GX(THICK)),OR,CL
,0
2460  RETURN
2470 '
2480 *LINE1
2490  IF CX=LPX AND CY=LPY GOTO *LRET
2500  IF HF=0 THEN GOSUB *CURSOR : GOSUB *LP
2510  IF NOT(THICK=1 AND MODE$='line') GOTO *LINE2
2520  LINE (CX,CY)-(LPX,LPY),CL
2530  GOTO *LRET
2540 '
2550 *LINE2
2560  X=SQR((CX-LPX)^2+(CY-LPY)^2)
2570  XS=(CX-LPX)/X : YS=(CY-LPY)/X
2580  DX=LPX : DY=LPY
2590 '
2600  IF MODE$='line' THEN R=THICK+1 ELSE R=THICK*3
2610  FOR I=0 TO X/R
2620    GOSUB *DOT
2630    IF INKEY$=CHR$(27) GOTO *LRET
2640    DX=DX+XS*R : DY=DY+YS*R
2650  NEXT I
2660 '
2670 *LRET
2680  IF F=-1 THEN GOSUB *LP : GOSUB *LPTOCR : F=0 : GOSUB *LP
2690  IF HF=0 THEN GOSUB *CURSOR : GOSUB *LP
2700  RETURN
2710 '
```

(DX, DY)に点を打つ

(DX,DY)にブラシの点を打つ

(LPX, LPY)から(CX, CY)まで線を引く

```
2720 *CIRCLEN
2730  IF CX=LPX AND CY=LPY GOTO *CNRET
2740  R=SQR((CX-LPX)^2+((CY-LPY)*2)^2)
2750  IF NOT(THICK=1 AND MODE$="line") GOTO *CIRCLE1
2760  GOSUB *CURSOR : GOSUB *LP
2770  CIRCLE (LPX,LPY),R,CL
2780  GOSUB *CURSOR : GOSUB *LP
2790  GOTO *CNRET
2800 *CIRCLE1
2810  XS=3.14159/180/(R/150)
2820  FOR I=0 TO 2*3.14159 STEP XS
2830    DX=LPX+R*COS(I) : DY=LPY-R/2*SIN(I)
2840    GOSUB *DOT
2850    IF INKEY$=CHR$(27) GOTO *CNRET
2860  NEXT I
2870 *CNRET
2880  RETURN
2890 '
2900 *PAINTER
2910  CLS : LOCATE 0,IY: COLOR IC
2920  INPUT "Paint  1:Normal , 2:Tile , 3:Return (1-3)";I
2930  IF I<1 OR I>3 GOTO *PAINTER
2940  ON I GOTO *NPAINT,*TILEPAINT : GOSUB *INFORM : RETURN
2950 '
2960 *NPAINT
2970  CLS : LOCATE 0,IY : COLOR IC
2980  INPUT "Paint color (0-7), Border color (1-7) : ";PC,BC
2990  IF PC>7 OR BC<1 OR BC>7 GOTO *NPAINT
3000  GOSUB *CURSOR : GOSUB *LP
3010  ON ERROR GOTO *ERRORSUB
3020  PAINT (CX,CY),PC,BC
3030  ON ERROR GOTO 0
3040  GOSUB *CURSOR : GOSUB *LP
3050  GOSUB *INFORM : RETURN
3060 '
3070 *TILEPAINT
3080  CLS : LOCATE 0,IY : COLOR IC
3090  INPUT "Paint color (Red,Green,Blue) : (0-4)";RC,GC,BC
3100  IF RC>4 OR GC>4 OR BC>4 GOTO *PAINTER
3110  TILE$=CHR$(D1(BC))+CHR$(D2(RC))+CHR$(D3(GC))
3120 '
3130 *BC : CLS : LOCATE 0,IY : COLOR IC
3140  INPUT "Border color : (1-7)";BC
3150  IF BC<1 OR BC>7 GOTO *BC
3160 '
3170  GOSUB *CURSOR : GOSUB *LP
3180  ON ERROR GOTO *ERRORSUB
3190  PAINT (CX,CY),TILE$,BC
3200  GOSUB *CURSOR : GOSUB *LP
3210  ON ERROR GOTO 0
3220  GOSUB *INFORM
3230  RETURN
3240 '
```

円を描く (2720–2880)

ノーマルペイント (2960–3050)

タイルペイント (3070–3230)

```
3250 *SMOOTH
3260  MAX=0 : CLS : LOCATE 0,IY : COLOR IC : PRINT "Smoothing";
3270 *DINPUT
3280  GOSUB *KEYINPUT : GOSUB *TENKEY
3290    IF P9=127 THEN GOSUB *INFORM : RETURN
3300    IF P9=191 THEN MAX=MAX+1 : X(MAX)=CX : Y(MAX)=CY : PSET(C
X,CY),CL
3310    IF P1=127 AND MAX>=2 GOTO *SMOOTHLINE
3320    IF P1=127 AND MAX<2 THEN GOSUB *INFORM: RETURN
3330    IF MAX=10 GOTO *SMOOTHLINE
3340    LOCATE 0,IY : PRINT "No.";MAX;" <=10";
3350  GOTO *DINPUT
3360 '
3370 *SMOOTHLINE : CLS
3380  IF MAX<3 THEN RETURN
3390  GOSUB *SMOOTHSUB
3400  X1=CX : Y1=CY : X2=LPX : Y2=LPY
3410  GOSUB *CURSOR : GOSUB *LP
3420  SLPX=LPX : SLPY=LPY : HF=-1
3430  FOR II=1 TO MAX-1
3440    LPX=X(II)  : LPY=Y(II)
3450    CX=X(II+1) : CY=Y(II+1)
3460    GOSUB *LINE1
3470  NEXT II
3480  LPX=SLPX : LPY=SLPY : HF=0
3490  GOSUB *CURSOR : GOSUB *LP
3500  CX=X1 : CY=Y1 : LPX=X2 : LPY=Y2
3510  GOSUB *INFORM
3520  RETURN
3530 '
3540 *SMOOTHSUB
3550  FOR II=1 TO 3
3560    N=0
3570    FOR I=2 TO MAX
3580      IF X(I)=X(I-1) AND Y(I)=Y(I-1) THEN N=N+1
3590      X(I-N)=X(I): Y(I-N)=Y(I)
3600    NEXT I
3610    MAX=MAX-N
3620    '
3630    FOR I=1 TO MAX-1
3640      D(I)=SQR((X(I)-X(I+1))^2+(Y(I)-Y(I+1))^2)
3650    NEXT I
3660    '
3670    MX(1,0)=(X(1)+X(2))/2 : MY(1,0)=(Y(1)+Y(2))/2
3680    N=1 : GOSUB *M1
3690    FOR I=2 TO MAX-2
3700      N=I : GOSUB *M0
3710      N=I : GOSUB *M1
3720    NEXT I
3730    N=MAX-1 : GOSUB *M0
3740    MX(MAX-1,1)=(X(MAX-1)+X(MAX))/2
3750    MY(MAX-1,1)=(Y(MAX-1)+Y(MAX))/2
3760    '
3770    FOR I=1 TO MAX-1
```

スムージングする点を入力する

スムージングのための計算

```
3780        MX(I,0)=(MX(I,0)+MX(I,1))/2
3790        MY(I,0)=(MY(I,0)+MY(I,1))/2
3800      NEXT I
3810      '
3820      J=MAX*2-1
3830      FOR I=MAX TO 2 STEP -1
3840        X(J)=X(I) : Y(J)=Y(I) : J=J-1
3850        X(J)=MX(I-1,0) : Y(J)=MY(I-1,0) : J=J-1
3860      NEXT I
3870      MAX=MAX*2-1
3880      '
3890   NEXT II : RETURN
3900 '
3910 *M0
3920   MX(N,0)=(2*X(N)+D(N)/2*(X(N)-X(N-1))/D(N-1)+X(N+1))/3
3930   MY(N,0)=(2*Y(N)+D(N)/2*(Y(N)-Y(N-1))/D(N-1)+Y(N+1))/3
3940   RETURN
3950 '
3960 *M1
3970   MX(N,1)=(X(N)+D(N)/2*(X(N+1)-X(N+2))/D(N+1)+2*X(N+1))/3
3980   MY(N,1)=(Y(N)+D(N)/2*(Y(N+1)-Y(N+2))/D(N+1)+2*Y(N+1))/3
3990   RETURN
4000 '
4010 *GETDATA
4020   LINE (0,5)-(6,5),7       : LINE (12,5)-(19,5),7
4030   LINE (8,0)-(10,3),7,BF : LINE (8,7)-(10,10),7,BF   メインカーソルを作る
4040   GET (0,0)-(19,10),CR
4050   LINE (0,0)-(19,10),0,BF
4060   '
4070   LINE (0,5)-(6,5),5       : LINE (12,5)-(19,5),5
4080   LINE (9,0)-(9,3),5,BF  : LINE (9,7)-(9,10),5,BF
4090   GET (0,0)-(19,10),LP                             LPカーソルを作る
4100   LINE (0,0)-(19,10),0,BF
4110   '
4120   SCREEN 1,1,0,7
4130   RESTORE 4290
4140   FOR K=1 TO 3
4150     READ IMAX,GX(K),GY(K)
4160     FOR J=0 TO DN-1
4170       FOR I=1 TO IMAX
4180         R=RND(1)*2*3.14159 : L=RND(1)*GX(K)
4190         PSET (L*SIN(R)+GX(K),L/2*COS(R)+GY(K)),7      ブラシのパターンを作る
4200       NEXT I
4210       GX2=GX(K)*2: GY2=GY(K)*2
4220       GET (0,0)-(GX2,GY2),B(K,J*GX(K))..............ブラシのパターンを読み込む
4230       LINE (0,0)-(GX2,GY2),0,BF
4240       PUT (0,0),B(K,J*GX(K)),PRESET
4250       GET (0,0)-(GX2,GY2),BR(K,J*GX(K))
4260       LINE (0,0)-(GX2,GY2),0,BF
4270     NEXT J
4280   NEXT K
4290   DATA 5,3,1, 12,7,3, 30,11,5
4300   '
4310   CIRCLE (4,2),3,7     : PAINT (4,2),7
4320   GET (0,0)-(8,4),L2   : LINE (0,0)-(8,4),0,BF
4330   PUT (0,0),L2,PRESET : GET (0,0)-(8,4),L2R
4340   LINE (0,0)-(8,4),0,BF
4350   '
4360   CIRCLE (9,4),8,7     : PAINT (9,4),7              太い筆を作る
```

```
4370  GET (0,0)-(18,8),L3 : LINE (0,0)-(18,8),0,BF
4380  PUT (0,0),L3,PRESET : GET (0,0)-(18,8),L3R
4390  LINE (0,0)-(18,8),0,BF
4400  SCREEN 0,0 : RETURN
4410 '
4420 *TILE
4430  RESTORE *TILEDATA
4440  FOR I=0 TO 4
4450    READ D1(I),D2(I),D3(I)
4460  NEXT I
4470  RETURN
4480 '
4490 *TILEDATA
4500  DATA 0,0,0, &H88,&H44,&H22, &Haa,&Haa,&H55
4510  DATA &Hee,&Hdd,&Hbb, &Hff,&Hff,&Hff
4520 '
4530 *ERRORSUB
4540  RESUME NEXT
4550 '
4560 *GDSAVE
4570  GOSUB *CURSOR : GOSUB *LP
4580  CLS : LOCATE 0,IY : COLOR IC
4590  PRINT "ｼｭｳﾘｮｳ ｼﾏｽ｡ Graphic Data ｦ save ｼﾏｽｶ ? (y/n)";
4600  INPUT I$ : I=INSTR("YyNn",I$)
4610  IF I<1 OR I>4 GOTO 4580
4620  IF I=3 OR I=4 GOTO 4820
4630 '
4640  CLEAR
4650  INPUT "file name within 6 letters ";NA$
4660  IF LEN(NA$)>6 GOTO 4640
4670  PRINT "Wait a minute..."
4680 '
4690  LINE (0,0)-(639,11),0,BF
4700  LINE (0,189)-(639,199),0,BF
4710  LINE (0,12)-(23,188),0,BF
4720  LINE (617,12)-(639,188),0,BF
4730  LINE (24,12)-(616,188),7,B
4740 '
4750  DIM GD(3002)
4760  FOR X=0 TO 1
4770    FOR Y=0 TO 1
4780      GET (X*320,Y*100)-(X*320+319,Y*100+99),GD
4790      BSAVE "1:"+NA$+"."+STR$(X*2+Y),VARPTR(GD(0)),3003*4
4800    NEXT Y
4810  NEXT X
4820  CLS : PRINT "Good-bye..." : END
4830 '
4840 *GDLOAD: CLEAR
4850  INPUT "input file name ";NA$
4860  PRINT "Wait a minute..."
4870 '
4880  DIM GD(3002)
4890  FOR X=0 TO 1
4900    FOR Y=0 TO 1
4910      BLOAD "1:"+NA$+"."+STR$(X*2+Y),VARPTR(GD(0))
4920      PUT (X*320,Y*100),GD
4930    NEXT Y
4940  NEXT X
4950  CLEAR : GOTO *INIT
```

タイルペイントに使うパターンを読む (4420–4470)

プログラムを終了する際の処理 (4560–4820)

絵のデータをディスクからロードする (4840–4950)

クイック リファレンス

参考文献

〔1〕「PC-8801　グラフィックスのすべて」
工藤丈彦　他著
アスキー出版局

〔2〕「PC-9801　ガイドブック」
榊　正憲　他著
アスキー出版局

〔3〕「コンピュータ　グラフィックス」
DAVID F. ROGERS
J. ALAN ADAMS　共著
山口富士夫　訳
日刊工業新聞社

〔4〕Oh! FM別冊「FILES Oh! FM」
日本ソフトバンク

〔5〕「バイオリズムとはなにか」
田多井吉之介　著
講談社ブルーバックス

〔6〕「PC-Techknow 8801 Vol.1」
栗山浩一　他著
システムソフト／アスキー

協力

今津英世・金光雅夫・坂中二郎・秋山耕一

PC-8801 mk II グラフィック・ワークブック

1985年1月15日　初版発行
定価1,500円

桜田幸嗣・蓑島聡
発行者　塚本慶一郎
発行所　株式会社アスキー
〒107 港区南青山5－11－5住友南青山ビル5F
振　替　東京4-161144
電　話　03－486－7111（代表）

編集担当　竹山正寿・大江朋子
表紙担当　郷　啓子
印刷　株式会社加藤文明社印刷所

ISBN4-87148-767-9 C3055 ¥1500E

デザイン・シート

x　座　標

	0	40	80	120	160	200	240	280	320	360	400	440	480	520	560	600
0																
40																
80																
120																
160																

y　座　標

キャラクタコード表

コード 10進	キャラクタ	対応するキー		コード 16進
0	コントロール・キャラクタ		CTRL @	&H00
1			CTRL A	&H01
2			CTRL B	&H02
3			CTRL C	&H03
4			CTRL D	&H04
5			CTRL E	&H05
6			CTRL F	&H06
7			CTRL G	&H07
8			CTRL H	&H08
9		TAB	CTRL I	&H09
10			CTRL J	&H0A
11		SHIFT HOME	CTRL K	&H0B
12		CLR	CTRL L	&H0C
13		↵	CTRL M	&H0D
14			CTRL N	&H0E
15			CTRL O	&H0F
16			CTRL P	&H10
17			CTRL Q	&H11
18		SHIFT INS	CTRL R	&H12
19			CTRL S	&H13
20			CTRL T	&H14
21			CTRL U	&H15
22			CTRL V	&H16
23			CTRL W	&H17
24			CTRL X	&H18
25			CTRL Y	&H19
26			CTRL Z	&H1A
27		ESC		&H1B
28		→		&H1C
29		←		&H1D
30		↑		&H1E
31		↓		&H1F

コード 10進	キャラクタ	コード 16進
32		&H20
33	!	&H21
34	"	&H22
35	#	&H23
36	$	&H24
37	%	&H25
38	&	&H26
39	'	&H27
40	(	&H28
41	)	&H29
42	*	&H2A
43	+	&H2B
44	,	&H2C
45	-	&H2D
46	.	&H2E
47	/	&H2F
48	0	&H30
49	1	&H31
50	2	&H32
51	3	&H33
52	4	&H34
53	5	&H35
54	6	&H36
55	7	&H37
56	8	&H38
57	9	&H39
58	:	&H3A
59	;	&H3B
60	<	&H3C
61	=	&H3D
62	>	&H3E
63	?	&H3F

コード 10進	キャラクタ	コード 16進
64	@	&H40
65	A	&H41
66	B	&H42
67	C	&H43
68	D	&H44
69	E	&H45
70	F	&H46
71	G	&H47
72	H	&H48
73	I	&H49
74	J	&H4A
75	K	&H4B
76	L	&H4C
77	M	&H4D
78	N	&H4E
79	O	&H4F
80	P	&H50
81	Q	&H51
82	R	&H52
83	S	&H53
84	T	&H54
85	U	&H55
86	V	&H56
87	W	&H57
88	X	&H58
89	Y	&H59
90	Z	&H5A
91	[	&H5B
92	¥	&H5C
93	]	&H5D
94	^	&H5E
95	_	&H5F

コード 10進	キャラクタ	コード 16進
96		&H60
97	a	&H61
98	b	&H62
99	c	&H63
100	d	&H64
101	e	&H65
102	f	&H66
103	g	&H67
104	h	&H68
105	i	&H69
106	j	&H6A
107	k	&H6B
108	l	&H6C
109	m	&H6D
110	n	&H6E
111	o	&H6F
112	p	&H70
113	q	&H71
114	r	&H72
115	s	&H73
116	t	&H74
117	u	&H75
118	v	&H76
119	w	&H77
120	x	&H78
121	y	&H79
122	z	&H7A
123	{	&H7B
124	|	&H7C
125	}	&H7D
126	~	&H7E
127	DEL	&H7F

コード 10進	キャラクタ	コード 16進
128		&H80
129		&H81
130		&H82
131		&H83
132		&H84
133		&H85
134		&H86
135		&H87
136		&H88
137		&H89
138		&H8A
139		&H8B
140		&H8C
141		&H8D
142		&H8E
143		&H8F
144		&H90
145		&H91
146		&H92
147		&H93
148		&H94
149		&H95
150		&H96
151		&H97
152		&H98
153		&H99
154		&H9A
155		&H9B
156		&H9C
157		&H9D
158		&H9E
159		&H9F

コード 10進	キャラクタ	コード 16進
160		&HA0
161	。	&HA1
162	「	&HA2
163	」	&HA3
164	、	&HA4
165	・	&HA5
166	ヲ	&HA6
167	ァ	&HA7
168	ィ	&HA8
169	ゥ	&HA9
170	ェ	&HAA
171	ォ	&HAB
172	ャ	&HAC
173	ュ	&HAD
174	ョ	&HAE
175	ッ	&HAF
176	ー	&HB0
177	ア	&HB1
178	イ	&HB2
179	ウ	&HB3
180	エ	&HB4
181	オ	&HB5
182	カ	&HB6
183	キ	&HB7
184	ク	&HB8
185	ケ	&HB9
186	コ	&HBA
187	サ	&HBB
188	シ	&HBC
189	ス	&HBD
190	セ	&HBE
191	ソ	&HBF

コード 10進	キャラクタ	コード 16進
192	タ	&HC0
193	チ	&HC1
194	ツ	&HC2
195	テ	&HC3
196	ト	&HC4
197	ナ	&HC5
198	ニ	&HC6
199	ヌ	&HC7
200	ネ	&HC8
201	ノ	&HC9
202	ハ	&HCA
203	ヒ	&HCB
204	フ	&HCC
205	ヘ	&HCD
206	ホ	&HCE
207	マ	&HCF
208	ミ	&HD0
209	ム	&HD1
210	メ	&HD2
211	モ	&HD3
212	ヤ	&HD4
213	ユ	&HD5
214	ヨ	&HD6
215	ラ	&HD7
216	リ	&HD8
217	ル	&HD9
218	レ	&HDA
219	ロ	&HDB
220	ワ	&HDC
221	ン	&HDD
222	゛	&HDE
223	゜	&HDF

コード 10進	キャラクタ	コード 16進
224		&HE0
225		&HE1
226		&HE2
227		&HE3
228		&HE4
229		&HE5
230		&HE6
231		&HE7
232	♠	&HE8
233	♥	&HE9
234	♦	&HEA
235	♣	&HEB
236	●	&HEC
237	○	&HED
238	╱	&HEE
239	╲	&HEF
240	╳	&HF0
241	円	&HF1
242	年	&HF2
243	月	&HF3
244	日	&HF4
245	時	&HF5
246	分	&HF6
247	秒	&HF7
248		&HF8
249		&HF9
250		&HFA
251		&HFB
252		&HFC
253		&HFD
254		&HFE
255		&HFF

ASCII
アスキーブックス

ISBN4-87148-767-9 C3055 ¥1500E 定価1,500円